INTERNATIONAL STANDARD

ISO 19444-1

Second edition
2019-08

Document management — XML Forms Data Format —

Part 1: Use of ISO 32000-2 (XFDF 3.0)

Gestion de documents — Format de Données des Formulaires XML — Partie 1: Utilisation de l'ISO 32000-2 (XFDF 3.0)

Reference number
ISO 19444-1:2019(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Fax: +41 22 749 09 47
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 171, *Document management applications*, Subcommittee SC 2, *Document file formats, EDMS systems and authenticity of information*.

This second edition cancels and replaces the first edition (ISO 19444-1:2016), which has been technically revised. The main changes compared to the previous edition are as follows:

— Addition of 3D comment related elements and attributes;

— Stream encoding information for XFDF.

A list of all parts in the ISO 19444 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

This document describes the XML Forms Data Format, which is used to represent form data from PDF (ISO 32000-2) in an XML tagset.

This format is derived from the forms data format in PDF and is intended to be a more interchangeable format for forms data.

Document management — XML Forms Data Format —

Part 1:
Use of ISO 32000-2 (XFDF 3.0)

1 Scope

This document specifies an XML format for representing forms data and annotations in the Portable Document Format, ISO 32000-2 (PDF 2.0).

This document does not change or add any definitions for any components of ISO 32000-2.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 32000-2:2017, *Document management — Portable Document Format — Part 2: PDF 2.0*

Extensible Markup Language (XML) 1.0. Fifth Edition. World Wide Web Consortium (W3C), November 2008

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 32000-2 apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

4 Notation

All XFDF element names, XFDF attributes names and examples (XFDF and PDF) are written in fixed width font. All inline PDF and FDF operators, keywords and the names of keys in PDF dictionaries are written in bold font. All inline words that denote operands of PDF and FDF operators or values of PDF dictionary keys are written in italic sans serif font.

Ellipses (...) are used within XFDF examples to indicate omitted detail.

5 Overview of XFDF

5.1 General

XFDF (XML Forms Data Format) is a format for representing forms data and annotations in a PDF document. This specification describes XFDF compatible with ISO 32000-2. XFDF is the XML version of Forms Data Format (FDF), a simplified version of PDF for representing forms data and annotations.

5.2 Forms data and annotations

Form fields in a PDF document may include many interactive elements (see ISO 32000-2:2017, 12.7, Forms). Figure 1 shows a sample PDF which uses edit boxes, buttons and radio buttons.

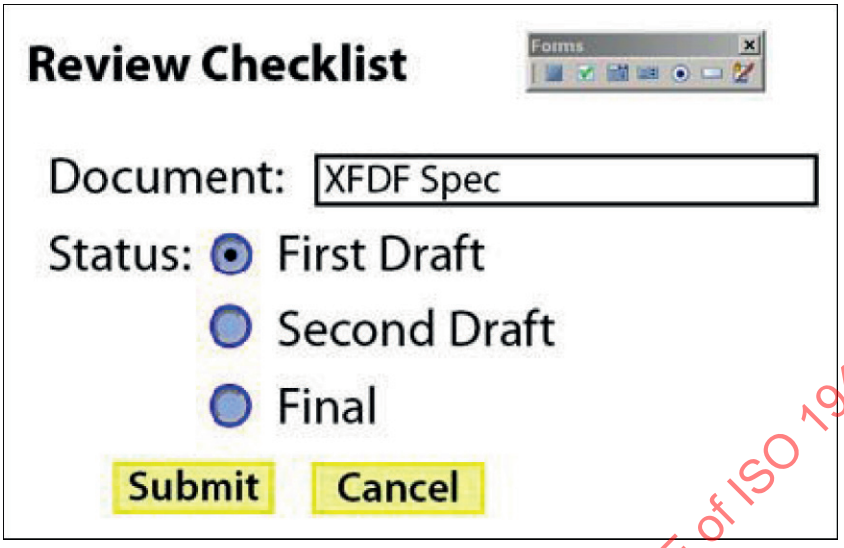


Figure 1 — Sample PDF utilising edit boxes, buttons and radio buttons

The XFDF exported from this PDF document might look like the following example:

```
<?xml version="1.0" encoding="UTF-8"?>

<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">

    <f href="Checklist.pdf"/>

    <ids original="7A0631678ED475F0898815F0A818CFA1"

    modified="BEF7724317B311718E8675B677EF9B4E"/>

    <fields>

        ...

    </fields>

</xfdf>
```

XFDF is often used as the format to send and receive PDF form data from a server. Form data is submitted to a server, modifications are made and sent back; the new form data is imported into the interactive PDF form. XFDF is also often used as a format to export form data to stand-alone files that can be stored, transmitted electronically and imported back into the corresponding PDF interactive form.

Annotations (see ISO 32000-2:2017, 12.5, “Annotations”) are attached to a PDF document. Figure 2 shows an example PDF, which includes text notes, highlights, stamps and file attachments.

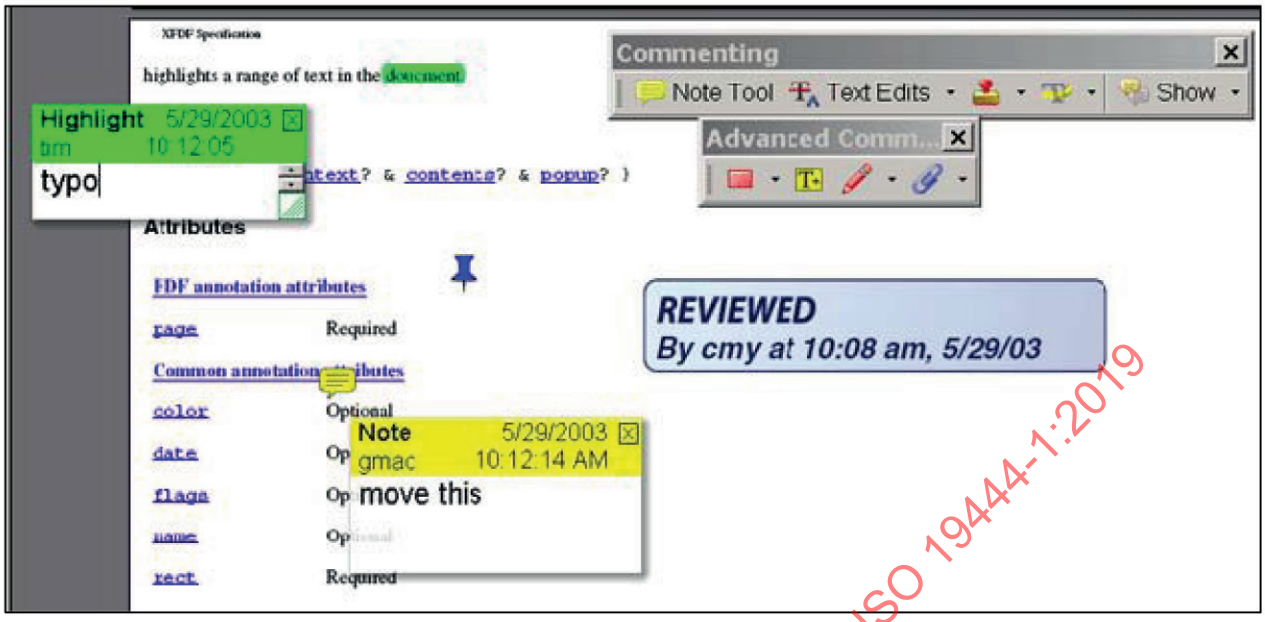


Figure 2 — Example PDF shown with text notes, highlights, stamps and file attachments

5.3 How to use this specification

This specification documents the correspondence between an XFDF element or attribute and its equivalent PDF dictionary and key. A short description is provided for each element and attribute; for complete information, see the description of the corresponding dictionary and key in ISO 32000-2. There are also a few attributes that do not correspond to a specific PDF dictionary and key.

5.4 PDF, FDF and XFDF

5.4.1 General

PDF, FDF and XFDF are related specifications with PDF as the parent format for representing documents, including interactive forms and annotations. FDF and XFDF contain the subset of a PDF document that describes interactive forms and annotations. Complete information on PDF and FDF can be found in ISO 32000-2.

FDF is a simplified version of PDF. PDF and FDF represent information with a key/value pair, also referred to as an entry. This example shows the **T** and **V** keys with values enclosed in parentheses:

```
/T (Street) /V (345 Park Ave.)
```

XFDF, on the other hand, represents an entry with an XML element/content or attribute/value pair, as shown in the corresponding XFDF:

```
<field name="Street">
   <value>345 Park Ave.</value>
</field>
```

XFDF implements a subset of FDF containing forms and annotations. There are XFDF equivalents for the **Annots**, **Fields**, **F** and **ID** keys of the FDF dictionary. There are no XFDF equivalents for the other entries in the FDF dictionary such as the **Status**, **Encoding**, **JavaScript**, **EmbeddedFDFs**, **Differences**, **Target** and **Pages** keys.

XFDF conforms to the XML standard.

In the simplest case, an XFDF element or attribute maps directly to a key in a particular dictionary of PDF. For example, the `creationdate` attribute is documented as corresponding to the **CreationDate** key in the markup annotation dictionary. This specification provides a description of the `creationdate` attribute, but more detailed information can be found in ISO 32000-2.

An example creationdate attribute in XFDF:

```
creationdate="D:20030425095243-07'00"
```

An equivalent entry in PDF or FDF would look like:

```
/CreationDate (D:20030425095243-07'00)
```

In the above example the underlying data stored within XFDF or PDF is identical. However, in other cases, the name and value differ. For example, the `flags` attribute corresponds to the **F** key in the annotation dictionary. The value of the `flags` attribute is a comma-separated list of the descriptive names of the flags, while the value of the **F** key is an integer with each bit representing a flag.

This is the XFDF flags attribute:

```
flags="print,nozoom,norotate"
```

This is the equivalent F entry in PDF or FDF:

```
/F 28
```

Finally, an element with multiple attributes can map to a single key with multiple values. The `ids` element in XFDF has attributes original and modified that map to the **ID** key in the FDF dictionary.

Below is an example `ids` element in XFDF:

```
<ids original="7A0631678ED475F0898815F0A818CFA1"

modified="BEF7724317B311718E8675B677EF9B4E" />
```

This is the corresponding ID entry in FDF:

```
/ID

[<7a0631678ed475f0898815f0a818cfa1><bef7724317b311718e8675b677ef9b4e

>]
```

5.4.2 Sample form in FDF and XFDF

Both FDF and XFDF for forms contain the same information: field name and value. In this FDF example, with line returns added for readability, the **Fields** key contains two fields named *Street* and *City*:

```
%FDF-1.2

%âãÏÓ

1 0 obj

<< /FDF

    << /F (Document.pdf)

       /ID [ <7a0631678ed475f0898815f0a818cfa1>

             <bef7724317b311718e8675b677ef9b4e> ]

       /Fields [
```

```
            << /T (Street)
               /V (345 Park Ave.) >>
            << /T (City) /V (San Jose)>>
         ] >>
>>
endobj
trailer
<< /Root 1 0 R >>
%%EOF
```

This is the XFDF version of the same form fields. The `fields` element contains two field elements with attribute name set to `Street` and `City`:

```
<?xml version="1.0" encoding="UTF-8"?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">
<f href="Document.pdf"/>
<ids original="7A0631678ED475F0898815F0A818CFA1"
modified="BEF7724317B311718E8675B677EF9B4E"/>
<fields>
   <field name="Street">
      <value>345 Park Ave.</value>
   </field>
   <field name="City">
      <value>San Jose</value>
   </field>
</fields>
</xfdf>
```

5.4.3 Sample annotation in FDF and XFDF

XFDF and FDF contain similar information but XFDF is represented in the XML format. This is a snippet of an FDF file containing a note annotation (line breaks added for readability):

```
%FDF-1.2
%âãÏÓ
1 0 obj
<< /FDF
   << /F (/C/Samples/Document.pdf)
      /ID [ <7a0631678ed475f0898815f0a818cfa1>
```

```
            <bef7724317b311718e8675b677ef9b4e> ]
      /Annots [4 0 R 3 0 R] >>
>>
endobj

3 0 obj
<< ... >>
endobj

4 0 obj
<< /F 28
   /Page 0
   ...
   /Type /Annot
   /Subj (Note)
   /Rect [271.850464 690.255371 291.850464 708.255371]
   /CreationDate (D:20030425095243-07'00')
   /NM (apYVRecPEj75sYIwSxME7C)
   ...
   /Subtype /Text
   ... >>
endobj
trailer
<< /Root 1 0 R >>
%%EOF
```

This is the same data in XFDF format:

```
<?xml version="1.0" encoding="UTF-8"?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">
 <f href="Document.pdf"/>
 <ids original="7A0631678ED475F0898815F0A818CFA1"
modified="BEF7724317B311718E8675B677EF9B4E"
/>
 <annots>
```

```
    <text flags="print,nozoom,norotate" page="0" subject="Note"
rect="271.850464,690.255371,291.850464,708.255371"
creationdate="D:20030425095243-07'00'"
name="apYVRecPEj75sYIwSxME7C" ...
>
        ...
        <popup .../>
    </text>
 </annots>
</xfdf>
```

5.5 Writing XFDF

5.5.1 General

This section describes XML implementation details specific to XFDF.

5.5.2 Encoding and namespace

The encoding in the XFDF file shall be UTF-8. Each XFDF file shall begin with the line:

```
<?xml version="1.0" encoding="UTF-8"?>
```

The namespace for XFDF shall be:

```
http://ns.adobe.com/xfdf/
```

The XML shall always specify that space is preserved in the XFDF files as follows:

```
xml:space="preserve"
```

Therefore, an XFDF document shall always begin with the following two lines:

```
<?xml version="1.0" encoding="UTF-8"?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">
```

5.6 Understanding forms

5.6.1 General

An XFDF file with form data contains form field names and values. When importing XFDF into a PDF processor, the target PDF file shall already contain the form fields. Importing XFDF updates the form field values in the PDF file. Exporting to XFDF puts the current value of the field in the `value` element.

Using XFDF, it is not possible to create a new form field in a PDF document or change anything other than the value of an existing form field.

5.6.2 Simple XFDF form

Figure 3 depicts a simple PDF document representing an address label containing text box form fields named *Name*, *Street* and *CityState*.

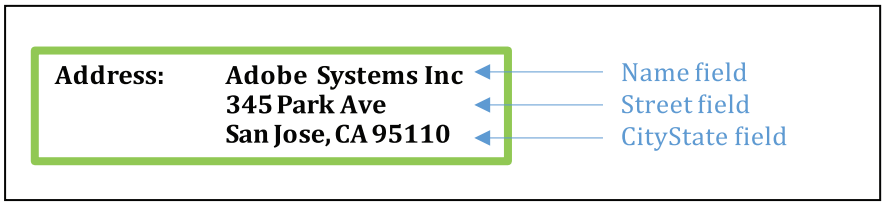


Figure 3 — A PDF representing an address label showing example form field names

In the example below, the `href` attribute on the f element points to the PDF document that contains the form fields. The `ids` element's original attribute contains a permanent identifier for the file, and the modified attribute contains an identifier that changes with each modification to the file. The `fields` element contains the three form fields and their respective values.

```
<?xml version="1.0" encoding="UTF-8"?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">
 <f href="samples/AddressLabel.pdf"/>
 <ids original="7A0631678ED475F0898815F0A818CFA1"
modified="BEF7724317B311718E8675B677EF9B4E"/>
 <fields>
    <field name="Name">
       <value>Adobe Systems, Inc.</value>
    </field>
    <field name="Street">
       <value>345 Park Ave.</value>
    </field>
    <field name="CityState">
       <value>San Jose, CA 95110</value>
    </field>
 </fields>
</xfdf>
```


5.6.3 Hierarchical XFDF form

Hierarchical form fields are commonly represented using a dot notation. If *Name*, *Street* and *CityState* are part of an Address, the fields are named:

```
Address.Name Address.Street Address.CityState
```

Figure 4 shows the same PDF rendition as Figure 3, but the field names are changed to hierarchical form fields.

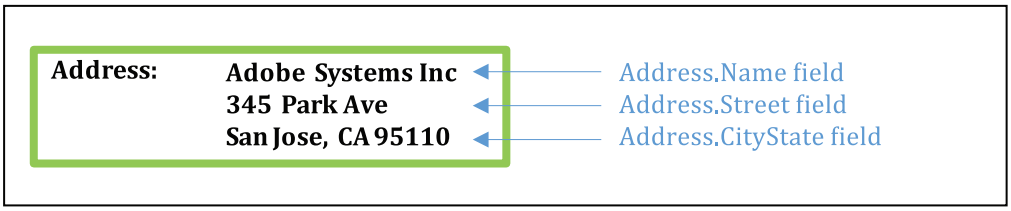


Figure 4 — A PDF representing an address label showing example hierarchical form field names

In XFDF exported from this PDF file, hierarchical form fields are represented using nested field elements. The Address field contains the Name, Street and CityState fields:

```
<?xml version="1.0" encoding="UTF-8"?>

<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">

 <fields>

    <field name="Address">

       <field name="Name">

          <value>Adobe Systems, Inc.</value>

       </field>

       <field name="Street">

          <value>345 Park Ave.</value>

       </field>

       <field name="CityState">

          <value>San Jose, CA 95110</value>

       </field>

    </field>

 </fields>

</xfdf>
```


5.7 Understanding annotations

5.7.1 General

XFDF annotations contain full information to recreate the annotation in a PDF document, including size and position on the page, the open or closed state of the annotation, colour and attached comments. Unlike forms, a new annotation can be created when XFDF is imported into a PDF file. However, this means that the XFDF for annotations is more complex than for forms.

Markup and **Popup** annotations are represented in XFDF; there are only eight annotations that are not represented in XFDF. Each annotation is represented by an element: for example, a **Text** annotation is represented by the `text` element, and a **Polygon** annotation is represented by the `polygon` element. Table 1 lists annotations that are supported and unsupported by XFDF.

Table 1 — List of supported and unsupported annotations

Supported annotation types	Unsupported annotations
`Text`	`Movie`
`FreeText`	`Widget`
`Line`	`Screen`
`Square`	`PrinterMark`
`Circle`	`TrapNet`
`Polygon`	`RichMedia` (PDF 2.0)
`Polyline`	`3D` (PDF 2.0)
`Highlight`	`Watermark` (PDF 2.0)
`Underline`	
`Squiggly`	
`StrikeOut`	
`Stamp`	
`Caret`	
`Ink`	
`Popup`	
`FileAttachment`	
`Sound`	
`Link`	
`Redact`	
`Projection` (PDF 2.0)	

5.7.2 Simple XFDF annotation

Figure 5 shows an example where a stamp annotation has been applied to a page in a PDF file.

Figure 5 — Stamp annotation applied to a PDF file

In the example below, the `href` attribute on the `f` element contains the name of the PDF file that exported the annotations. The `ids` element's original attribute contains a permanent identifier for the file, and the modified attribute contains an identifier that changes with each modification to the file.

Next is the `annots` element, which contains all annotations in the document. In this case, there is only one `stamp` annotation. In contrast to forms, annotations have many attributes, such as `color` or `title`, that can be modified and imported back into the PDF file to change the look of the annotation.

The `stamp` element contains a `popup` element which corresponds to the popup window for adding comments that is attached to the annotation. In this example, the popup window is empty and closed (`open="no"`).

```
<?xml version="1.0" encoding="UTF-8"?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">
 <f href="SimpleAnnot.pdf"/>
 <ids original="7A0631678ED475F0898815F0A818CFA1"
modified="BEF7724317B311718E8675B677EF9B4E"/>
 <annots>
    <stamp flags="print" page="0" subject="Approved"
rect="54.987381,671.039063,216.486893,718.539551"
creationdate="D:20030528192526-07'00'" name="jNrKlQf-J0kz3Y3a0cPjzA"
icon="SBApproved" color="#FF0000" date="D:20030528192529-07'00"
title="cmy">
       <popup flags="print,nozoom,norotate" page="0"
rect="612.000000,619.065979,792.000000,739.065979"
open="no"/>
    </stamp>
 </annots>
</xfdf>
```

5.7.3 Annotation with popup text

Figure 6 shows an example of the rubber stamp annotation with an open popup note with text.

Figure 6 — Example of a rubber stamp annotation with an open popup note

In the exported XFDF for the stamp element, the text of the popup is contained in a `contents-richtext` element which contains elements that conform to a subset of the XFA Text Specification. These are commonly referred to as rich text strings. For more information on rich text strings see 5.8.3, "Rich text strings". Here is the new `stamp` element with some attributes removed for readability:

```
<?xml version="1.0" encoding="UTF-8"?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">
 <stamp page="0" subject="Approved" icon="SBApproved" title="cmy">
```

```
    <contents-richtext>
       <body xmlns="http://www.w3.org/1999/xhtml"
xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/"
xfa:APIVersion="3.1.9078.0" xfa:spec="3.1">
          <p>
             <span style="font-size:10.0pt">Final and ready for
publishing.</span>
          </p>
       </body>
    </contents-richtext>
    <popup page="0"
rect="264.527802,648.970642,369.027802,715.470642"
open="yes"/>
 </stamp>
</xfdf>
```

5.7.4 Annotation with comment

Annotations can have comments attached to them. In Figure 7, the rubber stamp annotation has one comment.

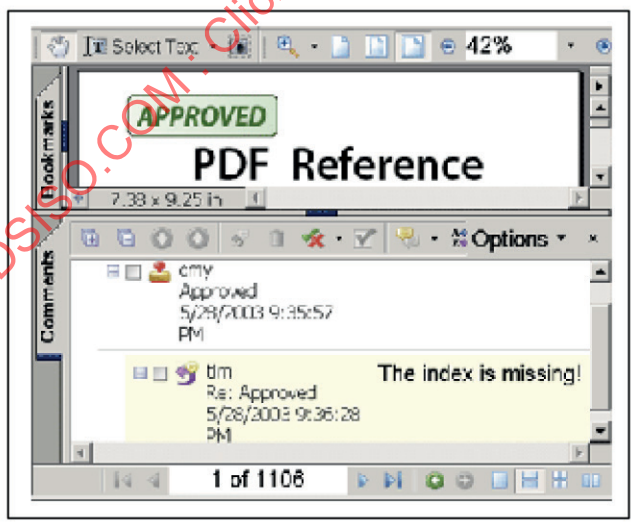


Figure 7 — Example of a rubber stamp annotation with one comment

The example below shows the `annots` element exported from the PDF file (attributes have been removed to improve readability). The comment is contained in the `text` element which is the second child of the `annots` element and follows the `stamp` element. The `text` element represents a comment because the value of the `inreplyto` attribute on `text` is identical to the value of the name attribute on stamp. The text of the annotation is contained in the `contents-richtext` element which is described in 5.8.3, Rich text strings.

```
<?xml version="1.0" encoding="UTF-8"?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">
 <annots>
    <stamp subject="Approved" name="HLjJ_qj5BC9dU1yKDfFD6D"
icon="SBApproved"
title="cmy">
      <popup open="no"/>
    </stamp>
    <text subject="Re: Approved" name="miAYuQ7A9JvIb3mFNkLjzC"
inreplyto="HLjJ_qj5BC9dU1yKDfFD6D" icon="Comment" title="tim">
      <contents-richtext>
          <body xmlns="http://www.w3.org/1999/xhtml"
xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/"
xfa:APIVersion="3.1.9078.0" xfa:spec="3.1">
             <p>
                <span style="font-size:10.0pt">The index is
missing!</span>
             </p>
          </body>
      </contents-richtext>
      <popup open="no"/>
    </text>
 </annots>
</xfdf>
```

5.8 Implementation notes

5.8.1 General

This section contains useful information regarding processing of the XFDF, as well as understanding the notation used in Clause 6.

5.8.2 String encoding conventions

XML requires that all content be in some particular character encoding. Much of PDF also has this requirement, but there are some strings in PDF for which the encoding is not known. In PDF these strings are designated as “string” and are effectively byte strings without any particular character interpretation.

The following convention is recommended for transforming these strings between PDF and XML:

— use ISO-Latin1 as the assumed encoding of the bytes in the PDF. For example, for the link annotation, this applies to the Named Destination name and the file `OriginalName`;

— escape any characters that are XML reserved or not legal code points in ISO-Latin1. Specifically, the escaping mechanism is:

  — if char is 0x0A, 0x0D or 0x09, emit

 or 	 respectively;

  — else if char < 0x20, emit escaped octal code (just like escaped sequences in PDF literal string). For example, code point 0x07 is emitted as \007;

  — else if char is 0x22, 0x26, 0x3C, 0x3E (XML delimiters), emit " & < or > respectively.

5.8.3 Rich text strings

Beginning with PDF 1.5, the text contents of variable text form fields and markup annotations can include formatting or style information. These rich text strings conform to a subset of the XFA Text Specification, which is itself a subset of the XHTML 1.0 specification, augmented with a restricted set of CSS2 style attributes. Rich text strings are fully described in ISO 32000-2.

Figure 8 shows a text field form that has a value formatted as rich text.

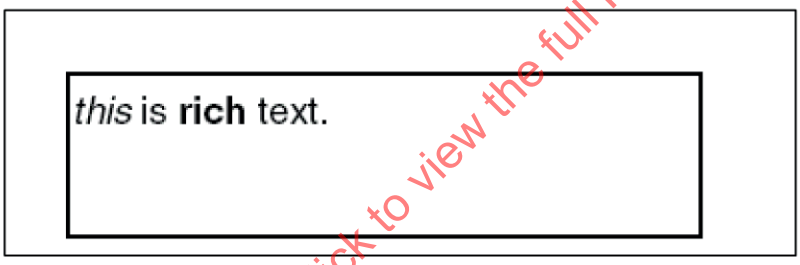


Figure 8 — Example of rich text

For example, the following text field form has a value formatted as rich text:

```
<?xml version="1.0" encoding="UTF-8"?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">
 <field name="myfield">
    <value-richtext>
       <body xmlns="http://www.w3.org/1999/xhtml"
xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/"
xfa:APIVersion="3.1.9078.0" xfa:spec="3.1">
          <p>
             <span style="font-size:10.0pt">
                <i>this</i> is <b>rich</b> text.</span>
          </p>
       </body>
```

```
        </value-richtext>
    </field>
</xfdf>
```

5.8.4 Stream encoding

The data of a stream is output to XML with two combinations of the mode and encoding attributes:

1. `mode="filtered" encoding="ascii"`

2. `mode="raw" encoding="hex"`

Method 1 may be used when the stream contains only printable ASCII characters, otherwise method 2 shall be used. Printable ASCII is where each byte of the stream when interpreted as an unsigned integer has a value less than 127 and greater than 32 or is a carriage return or linefeed. The XML control characters listed in Table 2 shall be converted (or filtered) to an entity.

Table 2 — XML control characters

Character	Entity
<	&lt;
>	&gt;
&	&amp;
"	&quot;

In method 2, the data is converted to a hexadecimal encoding where each byte is converted to a two character representation of the unsigned integer value, `[0-9A-F][0-9A-F]`. The high nibble shall always be first. For example, the ASCII space character is decimal 32 or hex 20.

5.8.5 XML content model syntax

5.8.5.1 Syntax overview

In the element reference (see 6.2, "XFDF elements"), a content model section is provided for each element. The content model defines the elements or types of text strings that can be contained by the element. For example, the content model for the `xfdf` element is:

```
( f? & ids? & fields? & annots? )
```

The content model is written using the symbols described in Table 1.

Table 3 — XML content model syntax

Symbol	Description
(	begin group
)	end group
,	followed by
&	and
|	or
?	0 or 1
+	1 or more
*	0 or more

The following are a few examples of content models:

5.8.5.2 Example 1

If element `lunch` can contain `salad` or `soup`, followed by `sandwich`, followed by an optional `dessert`, the content model is:

```
( salad | soup ), sandwich, dessert?
```

The following are valid `lunch` menus:

```
<lunch><salad/><sandwich/></lunch>

<lunch><soup/><sandwich/><dessert/></lunch>
```

However, the following is not a valid `lunch` because you cannot have both `salad` and `soup`:

```
<lunch><salad/><soup/><sandwich/></lunch>
```

The following is not valid because you shall have `salad` or `soup` and you cannot have two `dessert` elements:

```
<lunch><sandwich/><dessert/><dessert/></lunch>
```

5.8.5.3 Example 2

If element `sandwich` can contain, in any order, an optional `tomato` and optional `lettuce` element, the content model is:

```
( tomato? & lettuce? )
```

These are valid `sandwich` elements:

```
<sandwich><tomato/><lettuce/></sandwich>

<sandwich><lettuce/><tomato/></sandwich>

<sandwich/>

<sandwich><lettuce/></sandwich>

<sandwich><tomato/></sandwich>
```

This `sandwich` is not valid because it contains an extra `tomato`:

```
<sandwich><tomato/><lettuce/><tomato/><sandwich/>
```

6 XFDF reference

6.1 General

The following subclauses define a written schema for the elements and attributes that shall be used to represent an XFDF document. A conformant XFDF document shall contain only elements defined within this document. Each element shall contain only those elements defined within its content model and only those attributes specified for that respective element. All attributes specified as required shall be present if that element is used in the document and shall conform to the data types identified for that attribute.

6.2 XFDF elements

6.2.1 Elements

This subclause describes the top level `xfdf` element (6.2.2) and two of its children: (6.2.3) and ids (6.2.4).

6.2.2 xfdf

6.2.2.1 Overview

The `xfdf` element is the top level element in an XFDF document.

6.2.2.2 Content model

(f? & ids? & fields? & annots?)

6.2.2.3 Attributes

`xml:space` — *Required*. Value shall be `preserve`. This attribute in the xml namespace indicates that whitespace is preserved.

6.2.3 f

6.2.3.1 Overview

The f element is a child of the `xfdf` element and corresponds to the **F** key in the FDF dictionary. Specifies the source file or target file: the PDF document that this XFDF file was exported from or is intended to be imported into.

6.2.3.2 Content model

Empty

6.2.3.3 Attributes

`href` — *Required*. File specification (see ISO 32000-2:2017, 7.11, "File Specifications") pointing to the source file or target file.

6.2.4 ids

6.2.4.1 Overview

The `ids` element is a child of the `xfdf` element. The `ids` element corresponds to the **ID** Key in the FDF dictionary. The two attributes are file identifiers for the source or target file designated by the `f` element, taken from the **ID** entry in the PDF file's trailer dictionary.

6.2.4.2 Content model

Empty

6.2.4.3 Attributes

`original` — *Required*. This attribute corresponds to the permanent identifier which is based on the contents of the file at the time it was originally created. This value does not change when the file is incrementally updated.

The value shall be a hexadecimal number.

NOTE A common value for this is an MD5 checksum.

`modified` — *Required*. The `modified` attribute contains a unique identifier for the modified version of the PDF and corresponding XFDF document. The `modified` attribute corresponds to the changing identifier that is based on the file's contents at the time it was last updated.

The value shall be a hexadecimal number.

NOTE A common value for this is an MD5 checksum.

6.3 Form field elements

6.3.1 Fields

6.3.1.1 Overview

The `fields` element is a child of the `xfdf` element and is the container for form `field` elements. The `fields` element corresponds to the **Fields** key in the FDF dictionary.

6.3.1.2 Content model

*field**

6.3.1.3 Attributes

None

6.3.2 Field

6.3.2.1 Overview

The `field` element is a child of the `fields` and `field` elements. The `field` element corresponds to a form field.

6.3.2.2 Content model

(field | value* | (value? & value-richtext?))*

6.3.2.3 Attributes

`name` — *Required*. The name attribute corresponds to the **T** key in the FDF field dictionary.

In a hierarchical form field, the name is the partial field name.

6.3.2.4 Details

Hierarchical fields are represented by nesting `field` elements. In PDF, hierarchical fields are named with a dot notation: *phone.work* and *phone.home*. In XFDF, these are represented as:

```
<field name="phone">
```

```
    <field name="work"/>
    <field name="home"/>
  </field>
```

6.3.3 value

6.3.3.1 Overview

The `value` element is a child of the `field` element and contains the `field`'s value, for which the format may vary depending on the field type. The value corresponds to the **V** key in the FDF field dictionary.

A newline character in a PDF multi-line text field becomes a single line feed character in the contents of the value element.

Signature fields do not export a value.

6.3.3.2 Content model

Text string

6.3.3.3 Attributes

None

6.3.4 value-richtext

6.3.4.1 Overview

The `value-richtext` element is a child of the `field` element and contains the field's value formatted as a rich text string. This value corresponds to the **RV** key in the variable text field dictionary.

6.3.4.2 Content model

Text string or rich text string.

NOTE See 5.8.3, "Rich text strings".

6.3.4.3 Attributes

None

6.4 Annotation elements

6.4.1 annots

6.4.1.1 Overview

The `annots` element is a child of the `xfdf` element and serves as a container for annotation elements. The `annots` element corresponds to the **Annots** key in the FDF dictionary.

6.4.1.2 Content model

*(text | highlight | underline | strikeout | squiggly | line | circle | square | caret | polygon | polyline | stamp | ink | freetext | fileattachment | sound | link | redact | projection)**

6.4.1.3 Attributes

None

6.4.2 text

6.4.2.1 Overview

The `text` element is a child of the `annots` element and corresponds to a text annotation. A text annotation represents a “sticky note” attached to a page in the PDF document.

6.4.2.2 Content model

(contents-richtext? & contents? & popup?)

6.4.2.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation *attributes*

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*
`name`	*Optional*
`rect`	*Required*
`title`	*Optional*

Markup annotation attributes

`creationdate`	*Optional*
`opacity`	*Optional*
`subject`	*Optional*

Text annotation attributes

`icon`	*Optional*
`state`	*Optional*
`statemodel`	*Optional*
`inreplyto`	*Optional*
`replyType`	*Optional*

6.4.3 highlight

6.4.3.1 Overview

The `highlight` element is a child of the `annots` element and corresponds to the highlight Text annotation. A highlight annotation highlights a range of text in the document.

6.4.3.2 Content model

```
( contents-richtext? & contents? & popup? )
```

6.4.3.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation *attributes*

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*
`name`	*Optional*
`rect`	*Required*
`title`	*Optional*

Markup annotation attributes

`creationdate`	*Optional*
`opacity`	*Optional*
`subject`	*Optional*

Text annotation attributes

`coords`	*Required*

6.4.4 underline

6.4.4.1 Overview

The `underline` element is a child of the `annots` element and corresponds to the Underline Text Markup annotation. An Underline annotation appears as an underline in the text of the document.

6.4.4.2 Content model

```
( contents-richtext? & contents? & popup? )
```

6.4.4.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation *attributes*

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*
`name`	*Optional*
`rect`	*Required*
`title`	*Optional*

Markup annotation attributes

`creationdate`	*Optional*
`opacity`	*Optional*
`subject`	*Optional*

Text annotation attributes

`coords`	*Required*
`intent`	*Optional*

6.4.5 strikeout

6.4.5.1 Overview

The `strikeout` element is a child of the `annots` elements and corresponds to the Strikeout Text Markup annotation. A Strikeout annotation appears as a strikeout in the text of the document.

6.4.5.2 Content model

```
( contents-richtext? & contents? & popup? )
```

6.4.5.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation *attributes*

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*

`name` *Optional*

`rect` *Required*

`title` *Optional*

Markup annotation attributes

`creationdate` *Optional*

`opacity` *Optional*

`subject` *Optional*

Text annotation attributes

`coords` *Required*

6.4.6 squiggly

6.4.6.1 Overview

The `squiggly` element is a child of the `annots` element and corresponds to the Squiggly Text Markup annotation. The Squiggly annotation appears as a jagged underline in the text of a document.

6.4.6.2 Content model

(contents-richtext? & contents? & popup?)

6.4.6.3 Attributes

FDF annotation attributes

`page` *Required*

Common annotation *attributes*

`color` *Optional*

`date` *Optional*

`flags` *Optional*

`name` *Optional*

`rect` *Required*

`title` *Optional*

Markup annotation attributes

`creationdate` *Optional*

`opacity` *Optional*

`subject` *Optional*

Text annotation attributes

`coords` *Required*

6.4.7 line

6.4.7.1 Overview

The `line` element is a child of the `annots` element and corresponds to the Line annotation. A Line annotation displays a single straight line on the page.

6.4.7.2 Content model

```
( contents-richtext? & contents? & popup? )
```

6.4.7.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation *attributes*

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*
`name`	*Optional*
`rect`	*Required*
`title`	*Optional*

Markup annotation attributes

`creationdate`	*Optional*
`opacity`	*Optional*
`subject`	*Optional*

Text annotation attributes

`start`	*Required*
`end`	*Required*
`head`	*Optional*
`tail`	*Optional*
`interior-color`	*Optional*
`leaderLength`	*Optional*
`leaderExtended`	*Optional*
`caption`	*Optional*
`intent`	*Optional*
`leader-offset`	*Optional*

`caption-style`	*Optional*
`caption-offset-h`	*Optional*
`caption-offset-v`	*Optional*

Border style attributes

`width`	*Optional*
`dashes`	*Optional*
`style`	*Optional*

6.4.8 circle

6.4.8.1 Overview

The `circle` element is a child of the `annots` element and corresponds to the Circle annotation. A Circle annotation displays an ellipse on the page.

6.4.8.2 Content model

```
( contents-richtext? & contents? & popup? )
```

6.4.8.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation *attributes*

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*
`name`	*Optional*
`rect`	*Required*
`title`	*Optional*

Markup annotation attributes

`creationdate`	*Optional*
`opacity`	*Optional*
`subject`	*Optional*

Border style attributes

`width`	*Optional*
`dashes`	*Optional*
`style`	*Optional*

Border effect attributes

`intensity`	*Optional*
`style`	*Optional*

Circle and Square annotation attributes

`interior-color`	*Optional*
`fringe`	*Optional*

6.4.9 square

6.4.9.1 Overview

The `square` element is a child of the `annots` element and corresponds to the Square annotation. A Square annotation displays a rectangle on the page.

6.4.9.2 Content model

```
( contents-richtext? & contents? & popup? )
```

6.4.9.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation *attributes*

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*
`name`	*Optional*
`rect`	*Required*
`title`	*Optional*

Markup annotation attributes

`creationdate`	*Optional*
`opacity`	*Optional*
`subject`	*Optional*

Border style attributes

`width`	*Optional*
`dashes`	*Optional*
`style`	*Optional*

Border effect attributes

`intensity` *Optional*

`style` *Optional*

Circle and Square annotation attributes

`interior-color` *Optional*

`fringe` *Optional*

6.4.10 caret

6.4.10.1 Overview

The `caret` element is a child of the `annots` element and corresponds to the Caret annotation. A Caret annotation is a visual symbol that indicates the presence of text edits.

6.4.10.2 Content model

```
( contents-richtext? & contents? & defaultappearance? & popup? )
```

6.4.10.3 Attributes

FDF annotation attributes

`page` *Required*

Common annotation *attributes*

`color` *Optional*

`date` *Optional*

`flags` *Optional*

`name` *Optional*

`rect` *Required*

`title` *Optional*

Markup annotation attributes

`creationdate` *Optional*

`opacity` *Optional*

`subject` *Optional*

Caret annotation attributes

`fringe` *Optional*

`symbol` *Optional*

6.4.11 polygon

6.4.11.1 Overview

The `polygon` element is a child of the `annots` element and corresponds to the Polygon annotation. The Polygon annotation displays a closed polygon on the page.

6.4.11.2 Content model

```
( vertices & contents-richtext? & contents? & popup? )
```

6.4.11.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation *attributes*

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*
`name`	*Optional*
`rect`	*Required*
`title`	*Optional*

Markup annotation attributes

`creationdate`	*Optional*
`opacity`	*Optional*
`subject`	*Optional*

Border style attributes

`width`	*Optional*
`dashes`	*Optional*
`style`	*Optional*

Border effect attributes

`intensity`	*Optional*
`style`	*Optional*

Polygon and Polyline annotation attributes

`interior-color`	*Optional*
`intent`	*Optional*

6.4.12 polyline

6.4.12.1 Overview

The `polyline` element is a child of the `annots` element and corresponds to the Polyline annotation. The Polyline annotation is similar to the Polygon, but the first and last vertices are not connected. The polyline element has the same properties as polygon plus `LE` attributes.

6.4.12.2 Content model

```
( vertices & contents-richtext? & contents? & popup? )
```

6.4.12.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation *attributes*

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*
`name`	*Optional*
`rect`	*Required*
`title`	*Optional*

Markup annotation attributes

`creationdate`	*Optional*
`opacity`	*Optional*
`subject`	*Optional*

Border style attributes

`width`	*Optional*
`dashes`	*Optional*
`style`	*Optional*

Polygon and Polyline annotation attributes

`interior-color`	*Optional*
`head`	*Optional*
`tail`	*Optional*
`intent`	*Optional*

6.4.13 stamp

6.4.13.1 Overview

The `stamp` element is a child of the `annots` element and corresponds to the Rubber Stamp annotation. A Rubber Stamp annotation displays text or graphics intended to look as if they were stamped on the page with a rubber stamp.

If present, the `appearance` child element (the **AP** key in the annotation dictionary) shall take precedence over the icon attribute (**Name** key in the rubber stamp annotation dictionary).

6.4.13.2 Content model

```
( contents-richtext? & contents? & appearance? & popup? )
```

6.4.13.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation *attributes*

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*
`name`	*Optional*
`rect`	*Required*
`title`	*Optional*

Markup annotation attributes

`creationdate`	*Optional*
`opacity`	*Optional*
`subject`	*Optional*

Stamp annotation attributes

`icon`	*Optional*
`rotation`	*Optional*

6.4.14 ink

6.4.14.1 Overview

The `ink` element is a child of the `annots` element and corresponds to the ink annotation. An ink annotation represents a freehand "scribble" composed of one or more disjoint paths.

6.4.14.2 Content model

```
( inklist & contents-richtext? & contents? & popup? )
```

6.4.14.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation *attributes*

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*
`name`	*Optional*
`rect`	*Required*
`title`	*Optional*

Markup annotation attributes

`creationdate`	*Optional*
`opacity`	*Optional*
`subject`	*Optional*

Border style attributes

`width`	*Optional*
`dashes`	*Optional*
`style`	*Optional*

6.4.15 freetext

6.4.15.1 Overview

The `freetext` element is a child of the `annots` element and corresponds to the FreeText annotation. A FreeText annotation displays text directly on the page.

6.4.15.2 Content model

```
( defaultstyle? & contents-richtext? & contents? & defaultappearance )
```

6.4.15.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation *attributes*

`color`	*Optional*
`date`	*Optional*

`flags`	*Optional*
`name`	*Optional*
`rect`	*Required*
`title`	*Optional*

Markup annotation attributes

`creationdate`	*Optional*
`opacity`	*Optional*
`subject`	*Optional*

Border style attributes

`width`	*Optional*
`dashes`	*Optional*
`style`	*Optional*

Freetext annotation attributes

`rotation`	*Optional*
`justification`	*Optional*
`intent`	*Optional*

6.4.16 fileattachment

6.4.16.1 Overview

The `fileattachment` element is a child of the `annots` element and corresponds to a FileAttachment annotation. A FileAttachment annotation contains a reference to a file, which typically will be embedded in the PDF file.

6.4.16.2 Content model

```
( data & resource? & contents-richtext? & contents? )
```

6.4.16.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation *attributes*

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*
`name`	*Optional*

`rect` *Required*

`title` *Optional*

Markup annotation attributes

`creationdate` *Optional*

`opacity` *Optional*

`subject` *Optional*

Fileattachment annotation attributes

`icon` *Optional*

Embedded file parameter attributes

`size` *Optional*

`modification` *Optional*

`creation` *Optional*

`checksum` *Optional*

File specification attributes

`file` *Optional*

Miscellaneous attributes

`mimetype` *Optional*

6.4.17 sound

6.4.17.1 Overview

The `sound` element is a child of the `annots` element and corresponds to the sound annotation. A sound annotation is analogous to a text annotation, except that instead of a text note, it contains sound data.

6.4.17.2 Content model

```
( data & contents-richtext? & contents? )
```

6.4.17.3 Attributes

FDF annotation attributes

`page` *Required*

Common annotation *attributes*

`color` *Optional*

`date` *Optional*

`flags` *Optional*

`name` *Optional*

`rect` *Required*

`title` *Optional*

Markup annotation attributes

`creationdate` *Optional*

`opacity` *Optional*

`subject` *Optional*

Sound annotation attributes

`icon` *Optional*

`rate` *Required*

`bits` *Optional*

`channels` *Optional*

`encoding` *Optional*

6.4.18 link

6.4.18.1 Overview

The `link` element is a child of the `annots` element and corresponds to the link annotation. A link annotation identifies an area of the document where a link is to be available, and an action to perform or destination to go to should the link be activated.

6.4.18.2 Content model

```
( contents? & ( Dest | OnActivation ) & BorderStyleAlt? & popup? )
```

6.4.18.3 Attributes

FDF annotation attributes

`page` *Required*

Common annotation *attributes*

`color` *Optional*

`date` *Optional*

`flags` *Optional*

`name` *Optional*

`rect` *Required*

Border effect attributes

`style` *Optional*

Link annotation attributes

`Highlight`	*Optional*
`coords`	*Optional*

6.4.19 redact

6.4.19.1 Overview

The `redact` element is a child of the `annots` element and corresponds to the Redact annotation. A Redact annotation identifies content that is intended to be removed from the document. Redaction is a two-step process in which the user first applies redaction annotations that specify the pieces or regions of content that should be removed and subsequently instructs the PDF processor to apply the redact annotations and remove the content.

6.4.19.2 Content model

```
( contents-richtext? & contents? & popup? & defaultappearance? & overlayappearance? )
```

6.4.19.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation *attributes*

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*
`name`	*Optional*
`rect`	*Required*

Redaction annotation attributes

`coords`	*Optional*
`interior-color`	*Optional*
`overlay-text`	*Optional*
`overlay-text-repeat`	*Optional*
`justification`	*Optional*

6.4.20 projection

6.4.20.1 Overview

A projection annotation represents a comment made on a spatial model such as 3D artwork. The appearance of the annotation in a PDF page's two-dimensional coordinate system can be portrayed only as a projection of the comment's appearance in the spatial model. A projection annotation can be created, and its geometry and appearance can be modified, only within the context of an associated runtime environment, such as an activated 3D or geospatial model.

6.4.20.2 Content model

Empty

6.4.20.3 Attributes

FDF annotation attributes

`page`	*Required*

Common annotation *attributes*

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*
`name`	*Optional*
`rect`	*Required*
`title`	*Optional*

Markup annotation attributes

`creationdate`	*Optional*
`opacity`	*Optional*
`subject`	*Optional*

6.5 Annotation sub-elements

6.5.1 Action

6.5.1.1 Overview

The `Action` element is a child of the `OnActivation` sub-element of the link element and indicates an action for the interactive PDF processor to perform, such as launching an application or opening a new window. Corresponds to the **A** key in the annotation dictionary.

6.5.1.2 Content model

(URI | Launch | GoTo | GoToR | Named)

6.5.1.3 Attributes

None

6.5.2 appearance

6.5.2.1 Overview

The `appearance` element is a child of the stamp element and corresponds to the **AP** key in the annotation dictionary. The value is a base 64 encoded string.

6.5.2.2 Content model

Base 64 encoded string

6.5.2.3 Attributes

None

6.5.3 BorderStyleAlt

6.5.3.1 Overview

`BorderStyleAlt` is a child of the `link` element and corresponds to the **Border** key in the common annotation dictionary.

6.5.3.2 Content model

Border style encoded in the format specified in the border style attributes

6.5.3.3 Attributes

Border array attributes

`HCornerRadius`	*Required*
`VCornerRadius`	*Required*
`Width`	*Required*
`DashPattern`	*Optional*

6.5.3.4 Details

This format differs from the border style dictionary defined in the **BS** entry in the same table (represented in XDF by style, width and dashes). The **BS** style of border specification is more recently defined, but array-style borders are still prevalent.

6.5.4 contents

6.5.4.1 Overview

The `contents` element is a child of `caret`, `circle`, `fileattachment`, `freetext`, `highlight`, `ink`, `line`, `link`, `polygon`, `polyline`, `redact`, `sound`, `square`, `squiggly`, `stamp`, `strikeout`, `text` and `underline`.

This element corresponds to the common annotation key **Contents** in the annotation dictionary.

6.5.4.2 Content model

Text string

6.5.4.3 Attributes

None

6.5.4.4 Details

Text to be displayed for the annotation or, if this type of annotation does not display text, an alternate description of the annotation's contents in human-readable form. In either case, this text is useful when extracting the document's contents in support of accessibility to disabled users or for other purposes.

6.5.5 contents-richtext

6.5.5.1 Overview

The `contents-richtext` element is a child of `caret`, `circle`, `fileattachment`, `freetext`, `highlight`, `ink`, `line`, `polygon`, `polyline`, `redact`, `sound`, `square`, `squiggly`, `stamp`, `strikeout`, `text` and `underline`.

This element corresponds to the **RC** key in the markup annotation dictionary, which provides a rich text string to be displayed in the pop-up window when the annotation is opened.

6.5.5.2 Content model

Text string or rich text string.

See 5.8.3, Rich text strings.

6.5.5.3 Attributes

None

6.5.6 data

6.5.6.1 Overview

The `data` element is a child of the `fileattachment` and `sound` elements and contains the encoded file or sound data.

6.5.6.2 Content model

String encoded in the format specified in the mode and encoding attributes

6.5.6.3 Attributes

Miscellaneous attributes

`mode`	*Required*
`encoding`	*Required*

Miscellaneous attributes

`length`	*Required*
`filter`	*Optional*

6.5.6.4 Details

The stream data in the `data` element is output as described in subclause 5.8.4, "Stream encoding".

6.5.7 defaultappearance

6.5.7.1 Overview

The `defaultappearance` element is a child of the `caret` and `freetext` elements and corresponds to the **DA** key in the free text annotation dictionary (see 6.5.8, "defaultappearance" for the `defaultappearance` element that is a child of the `redact` element). The element specifies the default appearance string to be used in formatting the text.

6.5.7.2 Content model

Text string

6.5.7.3 Attributes

None

6.5.8 defaultappearance

6.5.8.1 Overview

The `defaultappearance` element is a child of the `redact` element and corresponds to the **DA** key in the redaction annotation dictionary (see 6.5.7, "defaultappearance" for the `defaultappearance` element that is a child of the `caret` and `freetext` elements). The value specifies the appearance string to be used in formatting the overlay text when it is drawn after the affected content has been removed. Its value shall be ignored if `overlayappearance` is present.

6.5.8.2 Content model

Text string

6.5.8.3 Attributes

None

6.5.9 defaultstyle

6.5.9.1 Overview

The `defaultstyle` element is a child of the `freetext` element and corresponds to the **DS** key in the free text annotation dictionary. A default style string.

6.5.9.2 Content model

Text string

6.5.9.3 Attributes

None

6.5.10 Dest

6.5.10.1 Overview

The `Dest` element is a child of the `link`, `GoTo` and `GoToR` elements and corresponds to the **Dest** key in the link annotations dictionary.

6.5.10.2 Content model

(Named | XYZ | Fit | FitH | FitV | FitR | FitB | FitBH | FitBV)

6.5.10.3 Attributes

None

6.5.10.4 Details

The target of the link is specified as a `name`, `string` or `array`.

6.5.11 File

6.5.11.1 Overview

The `File` element is a child of the `GoToR` and `Launch` elements and corresponds to the **F** key in the remote go-to actions and launch dictionaries.

6.5.11.2 Content model

None

6.5.11.3 Attributes

File specification attributes

`OriginalName`	*Required*

6.5.12 gesture

6.5.12.1 Overview

The `gesture` element is a child of the `inklist` element and contains the data from the **InkList** array.

6.5.12.2 Content model

Text string

6.5.12.3 Attributes

None

6.5.12.4 Details

The `gesture` element contains a `text string` made up of pairs of comma-separated real numbers separated by a semicolon. The pairs of real numbers represent a horizontal or vertical coordinate. Horizontal and vertical coordinate pairs represent a path. Therefore, the semicolon separated coordinates also occur in pairs.

Here is an example of the gesture element:

```
<gesture>87.712692,451.954437;85.805893,453.225616</gesture>
```

6.5.13 Fit

6.5.13.1 Overview

The `Fit` element is a child of the `Dest` element and corresponds to the **Fit** key in the destination syntax.

6.5.13.2 Content model

None

6.5.13.3 Attributes

Destination syntax attributes

`Page`	*Required*

6.5.13.4 Details

`Fit` displays the page designated by `Page`, with its contents magnified just enough to fit the entire page within the window both horizontally and vertically.

6.5.14 FitB

6.5.14.1 Overview

The `FitB` element is a child of the `Dest` element and corresponds to the **FitB** key in the destination syntax.

6.5.14.2 Content model

None

6.5.14.3 Attributes

Destination syntax attributes

`Page`	*Required*

6.5.14.4 Details

`FitB` displays the page designated by `Page`, with its contents magnified just enough to fit its bounding box entirely within the window both horizontally and vertically.

6.5.15 FitBH

6.5.15.1 Overview

The `FitBH` element is a child of the `Dest` element and corresponds to the **FitBH** key in the destination syntax.

6.5.15.2 Content model

None

6.5.15.3 Attributes

Destination syntax attributes

`Page`	*Required*
`Top`	*Required*

6.5.15.4 Details

`FitBH` displays the page designated by `Page`, with the vertical coordinate `Top` positioned at the top edge of the window and the contents of the page magnified just enough to fit the entire width of its bounding box within the window.

6.5.16 FitBV

6.5.16.1 Overview

The `FitBV` element is a child of the `Dest` element and corresponds to the **FitBV** key in the destination syntax.

6.5.16.2 Content model

None

6.5.16.3 Attributes

Destination syntax attributes

`Page`	*Required*
`Left`	*Required*

6.5.16.4 Details

`FitBV` displays the page designated by `Page`, with the horizontal coordinate `Left` positioned at the left edge of the window and the contents of the page magnified just enough to fit the entire height of its bounding box within the window.

6.5.17 FitH

6.5.17.1 Overview

The `FitH` element is a child of the `Dest` element and corresponds to the **FitH** key in the destination syntax.

6.5.17.2 Content model

None

6.5.17.3 Attributes

Destination syntax attributes

`Page`	*Required*
`Top`	*Required*

6.5.17.4 Details

`FitH` displays the page designated by `Page`, with the vertical coordinate `Top` positioned at the top edge of the window and the contents of the page magnified just enough to fit the entire width of the page within the window.

6.5.18 FitR

6.5.18.1 Overview

The `FitR` element is a child of the `Dest` element and corresponds to the **FitR** key in the destination syntax.

6.5.18.2 Content model

None

6.5.18.3 Attributes

Destination syntax attributes

`Page`	*Required*
`Left`	*Required*
`Bottom`	*Required*
`Right`	*Required*
`Top`	*Required*

6.5.18.4 Details

`FitR` displays the page designated by `Page`, with its contents magnified just enough to fit the rectangle specified by the coordinates `Left`, `Bottom`, `Right` and `Top` entirely within the window both horizontally and vertically.

6.5.19 FitV

6.5.19.1 Overview

The `FitV` element is a child of the `Dest` element and corresponds to the **FitV** key in the destination syntax.

6.5.19.2 Content model

None

6.5.19.3 Attributes

Destination syntax attributes

`Page`	*Required*
`Left`	*Required*

6.5.19.4 Details

`FitV` displays the page designated by `Page` with the horizontal coordinate `Left` positioned at the left edge of the window and the contents of the page magnified just enough to fit the entire width of the page within the window.

6.5.20 GoTo

6.5.20.1 Overview

The `GoTo` element is a child of the `Action` element and corresponds to the **GoTo** key in the action types dictionary.

6.5.20.2 Content model

Dest

6.5.20.3 Attributes

None

6.5.21 GoToR

6.5.21.1 Overview

The `GoToR` element is a child of the `Action` element and corresponds to the **GoToR** key in the action types dictionary.

6.5.21.2 Content model

(File & Dest)

6.5.21.3 Attributes

Remote go-to attributes

`NewWindow`	*Optional*

6.5.22 inklist

6.5.22.1 Overview

The `inklist` element is a child of the `ink` element and corresponds to the **InkList** key in the `Ink` annotation dictionary.

6.5.22.2 Content model

gesture+

6.5.22.3 Attributes

None

6.5.22.4 Details

The `inklist` element contains a series of gestures, each representing a stroked path. Each gesture is a series of alternating horizontal and vertical coordinates in default user space, specifying points along

the path. When drawn, the points are connected by straight lines or curves in an implementation-dependent way.

6.5.23 Launch

6.5.23.1 Overview

The `Launch` element is a child of the `Action` element and corresponds to the **Launch** key in the action types dictionary.

6.5.23.2 Content model

`File`

6.5.23.3 Attributes

Launch attributes

`NewWindow`	*Optional*

6.5.24 Named

6.5.24.1 Overview

The `Named` element is a child of the `Action` element and corresponds to the **Named** key in the action types dictionary.

6.5.24.2 Attributes

Named action attributes

`Name`	*Required*

6.5.25 Named

6.5.25.1 Overview

The `Named` element is a child of the `Dest` element and allows a destination to be referred to indirectly by means of a name object or a byte string.

6.5.25.2 Attributes

Destination syntax attributes

`Name`	*Required*

6.5.26 OnActivation

6.5.26.1 Overview

The `OnActivation` element is a child of the `link` element and corresponds to the `A` key in the link annotation dictionary.

6.5.26.2 Content model

`Action`

6.5.26.3 Attributes

None

6.5.27 overlayappearance

6.5.27.1 Overview

The `overlayappearance` element is a child of the `redact` element and corresponds to the **RO** key in the Redaction annotation dictionary. Its value is a form XObject specifying the overlay appearance for this redaction annotation. After this redaction is applied and the affected content has been removed, the overlay appearance should be drawn such that its origin lines up with the lower-left corner of the annotation rectangle. Takes precedence over the `interior-color`, `overlay-text`, `default-appearance` and `justification` attributes.

6.5.27.2 Content model

Text string

6.5.27.3 Attributes

None

6.5.28 popup

6.5.28.1 Overview

The `popup` element is a child of the `caret`, `circle`, `fileattachment`, `freetext`, `highlight`, `ink`, `line`, `link`, `polygon`, `polyline`, `redact`, `sound`, `square`, `squiggly`, `stamp`, `strikeout`, `text` and `underline` elements. This element corresponds to the Popup annotation which is described by the **Popup** key in the annotation dictionary. The popup annotation typically is associated with a parent annotation and is used for editing the parent's text.

6.5.28.2 Content model

Empty

6.5.28.3 Attributes

Common annotation *attributes*

`color`	*Optional*
`date`	*Optional*
`flags`	*Optional*
`name`	*Optional*
`rect`	*Required*
`title`	*Optional*

Popup annotation attributes

`open`	*Optional*

6.5.29 resource

6.5.29.1 Overview

As of PDF 2.0, this entry has been deprecated.

The `resource` element is a child of the `fileattachment` element and corresponds to the **ResFork** key in the Mac OS file information dictionary.

6.5.29.2 Content model

String encoded in the format specified in the mode and encoding attributes

6.5.29.3 Attributes

Miscellaneous attributes

`mode`	*Required*
`encoding`	*Required*

Stream *attributes*

`length`	*Required*
`filter`	*Required*

Mac OS file information attributes

`creator`	*Optional*
`subtype`	*Optional*

6.5.29.4 Details

The `resource` element contains the binary contents of the embedded file's resource fork. The data in the `resource` element is output as described in subclause 5.8.4, "Stream encoding".

6.5.30 URI

6.5.30.1 Overview

The `URI` element is a child of the `Action` element and corresponds to the **URI** key in the action types dictionary.

6.5.30.2 Content model

None

6.5.30.3 Attributes

URI attributes

`Name`	*Required*
`IsMap`	*Optional*

6.5.31 vertices

6.5.31.1 Overview

The `vertices` element is a child of the `polygon` and `polyline` elements and corresponds to the **Vertices** key in the polygon or polyline annotation dictionary.

6.5.31.2 Content model

Text string

6.5.31.3 Attributes

None

6.5.31.4 Details

An array of alternating horizontal and vertical coordinates of each vertex in default user space. The vertices element contains pairs of comma-separated real numbers representing a coordinate. Multiple pairs are separated by a semicolon.

6.5.32 XYZ

6.5.32.1 Overview

The `XYZ` element is a child of the `Dest` element and corresponds to the **XYZ** key in the destination syntax.

6.5.32.2 Content model

None

6.5.32.3 Attributes

Destination syntax attributes

`Page`	*Required*
`Left`	*Required*
`Bottom`	*Required*
`Right`	*Required*
`Top`	*Required*

6.6 Annotation attributes

All annotation attribute values are of XML string type. The annotation attributes are given in Tables 4 to 32.

Table 4 — FDF annotation attributes

Name	Description
`Page`	*Required*. The page attribute corresponds to the **Page** key in the FDF annotation dictionary. The `page` attribute represents the ordinal page number on which this annotation should appear, where page 0 is the first page. Elements: `caret`, `circle`, `fileattachment`, `freetext`, `highlight`, `ink`, `line`, `link`, `polygon`, `polyline`, `projection`, `redact`, `sound`, `square`, `squiggly`, `stamp`, `strikeout`, `text` and `underline`

Table 5 — Common annotation attributes

Name	Description
`color`	*Optional*. The `color` attribute corresponds to the **C** key. The **C** key contains an array of three numbers between 0.0 and 1.0 in the **DeviceRGB** colour space. Each colour is mapped to a value between 0 and 255 then converted to hexadecimal (00 to FF). The three hexadecimal values are concatenated and prefixed with a hash sign: `color="#FFFF00"` Elements: `caret`, `circle`, `fileattachment`, `freetext`, `highlight`, `ink`, `line`, `link`, `polygon`, `polyline`, `projection`, `redact`, `sound`, `square`, `squiggly`, `stamp`, `strikeout`, `text` and `underline`
`date`	*Optional*. Corresponds to the **M** Key. The preferred format is a PDF date string, but PDF processors should be prepared to display a string in any format. Elements: `caret`, `circle`, `fileattachment`, `freetext`, `highlight`, `ink`, `line`, `link`, `polygon`, `polyline`, `projection`, `redact`, `sound`, `square`, `squiggly`, `stamp`, `strikeout`, `text` and `underline`
`flags`	*Optional*. Default: is no flags. Corresponds to the **F** key. A set of flags specifying various characteristics of the field's widget annotation. The value is a comma-separated list containing zero or more of the following values: — `invisible` — `hidden` — `print` — `nozoom` — `norotate` — `noview` — `readonly` — `locked` — `togglenoview` Example: `flags="print,locked"` Elements: `caret`, `circle`, `fileattachment`, `freetext`, `highlight`, `ink`, `line`, `link`, `polygon`, `polyline`, `projection`, `redact`, `sound`, `square`, `squiggly`, `stamp`, `strikeout`, `text` and `underline`

Table 5 *(continued)*

Name	Description
`name`	*Optional*. Corresponds to the **NM** key. A string containing the annotation name, a text string uniquely identifying it among all the annotations on its page. Elements: `caret`, `circle`, `fileattachment`, `freetext`, `highlight`, `ink`, `line`, `link`, `polygon`, `polyline`, `projection`, `redact`, `sound`, `square`, `squiggly`, `stamp`, `strikeout`, `text` and `underline`
`rect`	*Required*. Corresponds to the **Rect** key. The annotation rectangle, defining the location of the annotation on the page in default user space units. The value is four comma-separated real numbers which may be positive or negative. Elements: `caret`, `circle`, `fileattachment`, `freetext`, `highlight`, `ink`, `line`, `link`, `polygon`, `polyline`, `projection`, `redact`, `sound`, `square`, `squiggly`, `stamp`, `strikeout`, `text` and `underline`
`title`	*Optional*. Corresponds to the **T** key. The text label to be displayed in the title bar of the annotation's popup window when open and active. Elements: `caret`, `circle`, `fileattachment`, `freetext`, `highlight`, `ink`, `line`, `polygon`, `polyline`, `projection`, `sound`, `square`, `squiggly`, `stamp`, `strikeout`, `text` and `underline`

Table 6 — Markup annotation attributes

Name	Description
`creationdate`	*Optional*. Corresponds to the **CreationDate** entry. The date and time when the annotation was created. Value is in PDF date format. NOTE The format for `creationdate` is intentionally different than that of `date` in 0, "Table 4 — Common annotation attributes". Elements: `caret`, `circle`, `fileattachment`, `freetext`, `highlight`, `ink`, `line`, `polygon`, `polyline`, `projection`, `sound`, `square`, `squiggly`, `stamp`, `strikeout`, `text`, `underline` and `projection`
`opacity`	*Optional*. Default is 1.0. Value is a decimal number. Corresponds to the **CA** key. The constant opacity value to be used in painting the annotation. This value applies to all visible elements of the annotation in its closed state (including its background and border), but not to the popup window that appears when the annotation is opened. The specified value is not used if the annotation has an appearance stream; in that case, the appearance stream itself shall specify any desired transparency. The implicit blend mode is `Normal`. Elements: `caret`, `circle`, `fileattachment`, `freetext`, `highlight`, `ink`, `line`, `polygon`, `polyline`, `sound`, `square`, `squiggly`, `stamp`, `strikeout`, `text`, `underline` and `projection`
`subject`	*Optional*. Corresponds to the **Subj** key. Text representing a short description of the subject being addressed by the annotation. The value is a string. Elements: `caret`, `circle`, `fileattachment`, `freetext`, `highlight`, `ink`, `line`, `polygon`, `polyline`, `projection`, `sound`, `square`, `squiggly`, `stamp`, `strikeout`, `text`, `underline` and `projection`
`intent`	*Optional*. A name describing the intent of the markup annotation. Corresponds to the **IT** key in the markup annotation dictionary. Intents allow PDF processors to distinguish between different uses and behaviours of a single markup annotation type. If this entry is not present or its value is the same as the annotation type, the annotation has no explicit intent and should behave in a generic manner in a PDF processor. Elements: `freetext`, `line`, `polygon` and `polyline`

Table 7 — Text markup annotation attributes

Name	Description
`coords`	*Required*. Corresponds to the **QuadPoints** key in the text markup annotation dictionary. Value is one or more groups of eight comma-separated real numbers. Groups are separated by commas. An array of 8 x n numbers specifying the coordinates of n quadrilaterals in default user space. Each quadrilateral encompasses a word or group of contiguous words in the text underlying the annotation. The coordinates for each quadrilateral are given in the following order: `x1,y1,x2,y2,x3,y3,x4,y4` These specify the quadrilateral's four vertices in counter clockwise order. The text is oriented with respect to the edge connecting points `(x1,y1)` and `(x2,y2)`. Elements: `highlight`, `squiggly`, `strikeout` and `underline`
`inreplyto`	*Required if* `replyType` *is present, otherwise optional*. Corresponds to the **IRT** key in the markup annotation dictionary. A reference to the annotation to which this annotation is in reply. Both annotations shall be on the same page of the document. The value is not a dictionary but a text string containing the contents of the name attribute of the annotation being replied to, to allow for a situation where the annotation being replied to is not in the same file. Elements: `text`
`replyType`	*Optional, only meaningful if* `inreplyto` *is present*. Default value is `reply`. A name specifying the relationship (the "reply type") between this annotation and the one specified by `inreplyto`. Corresponds to the **RT** key in the markup annotation dictionary. Values are: — `reply` (default) — `group` — `Elements:` text

Table 8 — Text annotation attributes

Name	Description
`icon`	*Optional*. The icon attribute corresponds to the **Name** key in the text annotation dictionary. The name of the icon to be used in displaying the annotation. PDF processors should provide predefined icon appearances for at least the following standard names:

Table 8 *(continued)*

Name	Description
	— `Comment` — `Check` — `Circle` — `Cross` — `Help` — `Insert` — `Key` — `NewParagraph` — `Note` (default) — `Paragraph` — `RightArrow` — `RightPointer` — `Star` — `UpArrow` — `UpLeftArrow` Additional names may be supported as well. Elements: `text`
`state`	*Optional*. The state to which the annotation should be set. The state attribute corresponds to the State key in the text annotation dictionary. If `statemodel` is set to `Marked`, the default value is `Unmarked`. If `statemodel` is set to `Review`, the default value is `None`. Values are: — `Marked` — `Unmarked` — `Accepted` — `Rejected` — `Cancelled` — `Completed` — `None` Elements: `text`
`statemodel`	*Required if* `state` *is present, otherwise optional*. The `statemodel` attribute corresponds to the **StateModel** key in the text annotation dictionary. Values are: — `Marked` — `Review` Elements: `text`

Table 9 — Line annotation attributes

Name	Description
`start`	*Required*. Two comma-separated real numbers specify the starting coordinates. Corresponds to the first two numbers in the **L** key in the line annotation dictionary. The **L** key is an array of four numbers specifying the starting and ending coordinates of the line in default user space. Elements: `line`
`end`	*Required*. Two comma-separated real numbers specify the starting coordinates. Corresponds to the first two numbers in the **L** key in the line annotation dictionary. The **L** key is an array of four numbers specifying the starting and ending coordinates of the line in default user space. Elements: `line`
`head`	*Optional*. Default: `None`. The line end for the head. Corresponds to first name in the **LE** key in the line annotation dictionary. The **LE** key is an array of two names specifying the line ending styles to be used in drawing the line. Values are: — `None` (default) — `Square` — `Circle` — `Diamond` — `OpenArrow` — `ClosedArrow` — `Butt` — `ROpenArrow` — `RClosedArrow` Elements: `line`

Table 9 *(continued)*

Name	Description
`tail`	*Optional*. Default: `None`. The line end for the tail. Corresponds to second name in the **LE** key in the line annotation dictionary. The **LE** key is an array of two names specifying the line ending styles to be used in drawing the line. Values are: — `None` (default) — `Square` — `Circle` — `Diamond` — `OpenArrow` — `ClosedArrow` — `Butt` — `ROpenArrow` — `RClosedArrow` Elements: `line`
`interior-color`	*Optional*. Corresponds to the **IC** key in the line annotation dictionary and specifies the interior colour with which to fill the annotation's line endings. If this entry is absent, the interiors of the line endings are left transparent. The **IC** key contains an array of three numbers between 0.0 and 1.0 in the **DeviceRGB** colour space. Each colour is mapped to a value between 0 and 255 then converted to hexadecimal (00 to FF). The three hexadecimal values are concatenated and prefixed with a hash sign. For example: `interior-color="#FFFF00"` Elements: `line`
`leaderLength`	*Required if* `LeaderExtend` *is present; otherwise optional*. Default: 0 (no leader lines). Corresponds to the **LL** key in the line annotation dictionary and specifies the length of leader lines in default user space that extend from each endpoint of the line perpendicular to the line itself. A positive value means that the leader lines appear in the direction that is clockwise when traversing the line from its starting point to its ending point (as specified by the **L** key); a negative value indicates the opposite direction. Elements: `line`
`leaderExtend`	*Optional*. Default: 0 (no leader line extensions). Value is a non-negative number. Corresponds to the **LLE** key in the line annotation dictionary and specifies the length of leader line extensions that extend from the line proper 180 degrees from the leader. Elements: `line`

Table 9 *(continued)*

Name	Description
`caption`	*Optional*. A flag specifying whether or not the text specified by the `contents` or `contents-richtext` entries should be replicated as a caption in the appearance of the line. Corresponds to the **Cap** key in the line annotation dictionary. The text should be rendered in a manner appropriate to the content, taking into account factors such as writing direction. Values: — `yes` — `no` (default) Elements: `line`
`intent`	*Optional*. A name describing the intent of the line annotation. Corresponds to the **IT** key in the line annotations dictionary. Values: — `LineArrow` — `LineDimension` Elements: `line`
`leader-offset`	*Optional*. Default: 0 (no leader line offset). A non-negative number representing the length of the leader line offset. Corresponds to the **LLO** key in the line annotations dictionary. The leader line offset is the amount of empty space between the endpoints of the annotation and where the leader lines begin. Elements: `line`
`caption-style`	*Optional*. Meaningful only if caption is yes. A name describing the annotation's caption style. Corresponds to the **CP** key in the line annotation dictionary. Values: — `Top` — `Inline` (default) Elements: `line`
`caption-offset-h`	*Optional*. Default: 0 (no offset). Meaningful only if caption is yes. A number specifying the horizontal offset of the caption text from its normal positioning. Corresponds to the first entry in the **CO** key array in the line annotation dictionary. The horizontal offset is measured along the annotation line from its midpoint, with a positive value indicating offset to the right and a negative value indicating offset to the left. Elements: `line`
`caption-offset-v`	*Optional*. Default: 0 (no offset). Meaningful only if caption is yes. A number specifying the vertical offset of the caption text from its normal positioning. Corresponds to the first entry in the **CO** key array in the line annotation dictionary. The vertical offset is measured perpendicular to the annotation line, with a positive value indicating a shift up and a negative value indicating a shift down. Elements: `line`

Table 10 — Circle and Square annotation attributes

Name	Description
`interior-color`	*Optional*. Corresponds to the **IC** key in the line annotation dictionary and specifies the interior colour with which to fill the annotation's line endings. If this entry is absent, the interiors of the line endings are left transparent. The **IC** key contains an array of three numbers between 0,0 and 1,0 in the **DeviceRGB** colour space. Each colour is mapped to a value between 0 and 255 then converted to hexadecimal (00 to FF). The three hexadecimal values are concatenated and prefixed with a hash sign. For example: `interior-color="#FFFF00"` Elements: `circle`, `square`
`fringe`	*Optional*. The fringe attribute corresponds to the **RD** key in the circle or square annotation dictionary and is a set of four values describing the numerical differences between two rectangles: the **Rect** entry of the annotation and the actual boundaries of the underlying object. Value is four comma-separated real numbers, representing left, top right and bottom differences. Elements: `circle`, `square`

Table 11 — Caret annotation attributes

Name	Description
`Symbol`	*Optional*. The symbol attribute corresponds to the **Sy** key in the caret annotation dictionary. Value is a name specifying a symbol to be associated with the caret: — `none` (default) — `paragraph` Elements: `caret`
`Fringe`	*Optional*. The `fringe` attribute is a rectangle that corresponds to the **RD** key in the circle or square annotation dictionary and is a set of four values describing the numerical differences between two rectangles: the **Rect** entry of the annotation and the actual boundaries of the underlying object. Value is four comma-separated real numbers, representing left, top right and bottom differences. Elements: `caret`

Table 12 — Polygon and Polyline annotation attributes

Name	Description
`interior-color`	*Optional*. Corresponds to the **IC** key in the line annotation dictionary and specifies the interior colour with which to fill the annotation's line endings. If this entry is absent, the interiors of the line endings are left transparent. The **IC** key contains an array of three numbers between 0.0 and 1.0 in the **DeviceRGB** colour space. Each colour is mapped to a value between 0 and 255 then converted to hexadecimal (00 to FF). The three hexadecimal values are concatenated and prefixed with a hash sign. For example: `interior-color="#FFFF00"` Elements: `polygon`, `polyline`

Table 12 *(continued)*

Name	Description
`Head`	*Optional*. Default: `None`. The line end for the head. Corresponds to first name in the **LE** key in the line annotation dictionary. The **LE** key is an array of two names specifying the line ending styles to be used in drawing the line. Values are: — `None` (default) — `Square` — `Circle` — `Diamond` — `OpenArrow` — `ClosedArrow` — `Butt` — `ROpenArrow` — `RClosedArrow` Elements: `polyline`
`tail`	*Optional*. Default: `None`. The line end for the tail. Corresponds to second name in the **LE** key in the line annotation dictionary. The **LE** key is an array of two names specifying the line ending styles to be used in drawing the line. Values are: — `None` (default) — `Square` — `Circle` — `Diamond` — `OpenArrow` — `ClosedArrow` — `Butt` — `ROpenArrow` — `RClosedArrow` Elements: `polyline`
`intent`	*Optional*. A name describing the intent of the line annotation. Corresponds to the **IT** key in the line annotations dictionary. Values: — `PolygonCloud` — `polygon-dimension` — `polyline-dimension` Elements: `polygon`, `polyline`

Table 13 — Freetext annotation attributes

Name	Description
`justification`	*Optional*. The justification attribute corresponds to the **Q** key in the free text annotation dictionary. It specifies the form of quadding (justification) to be used in displaying the annotation's text. Values are: — `left` (default) — `centered` — `right` Elements: `freetext`
`rotation`	*Optional*. Value is an integer. Corresponds to the **Rotate** key. An integer representing the clockwise rotation in degrees. Elements: `freetext`
`tail`	*Optional*. Value is a name describing the intent of the freetext annotation. Corresponds to the **IT** key in the freetext annotations dictionary. Values are: — `FreeTextCallout` — `FreeTextTypeWriter` Elements: `freetext`

Table 14 — Stamp annotation attributes

Name	Description
`icon`	*Optional*. Default: `Draft`. Corresponds to the **Name** key in the rubber stamp annotation dictionary. The name of an icon to be used in displaying the annotation.

Table 14 *(continued)*

Name	Description
	Common values are: — `SBRejected` — `SHAccepted` — `SHInitialHere` — `SHSignHere` — `SHWitness` — `SBApproved` — `SBCompleted` — `SBConfidential` — `SBDraft` — `SBFinal` — `SBForComment` — `SBForPublicRelease` — `SBInformationOnly` — `SBNotApproved` — `SBNotForPublicRelease` — `SBPreliminaryResults` — `SBVoid` — `Approved` — `AsIs` — `Confidential` — `Departmental` — `Draft` (default) — `Experimental` — `Expired` — `Final` — `ForComment` — `ForPublicRelease` — `NotApproved` — `NotForPublicRelease` — `Sold` — `TopSecret` Additional names may be supported as well. Elements: `stamp`

Table 14 *(continued)*

Name	Description
`rotation`	*Optional*. Value is an integer. Corresponds to the **Rotate** key. An integer representing the clockwise rotation in degrees. Elements: stamp

Table 15 — File attachment annotation attributes

Name	Description
`icon`	*Optional*. The icon attribute corresponds to the **Name** key in the file attachment annotation dictionary. The name of an icon to be used in displaying the annotation. PDF processors should provide predefined icon appearances for at least the predefined values. Additional names may be supported as well. Value may be a predefined value or a string. The predefined values are: — `Graph` — `Paperclip` — `PushPin` (default) — `Tag` Elements: `fileattachment`

Table 16 — Sound annotation attributes

Name	Description
`icon`	*Optional*. The icon attribute corresponds to the **Name** key in the sound annotation dictionary and is the name of an icon to be used in displaying the annotation. PDF processors should provide predefined icon appearances for at least the standard names; additional names may be supported as well. Values are: — `Speaker` (default) — `Mic` — `Ear` `Elements: sound`
`bits`	*Optional*. Default: `8`. Corresponds to the **B** key for a sound object and is an integer describing the number of bits per sample value per channel. Elements: `sound`
`bits`	*Optional*. Default: `1`. Corresponds to the **C** key for a sound object and is an integer describing the number of sound channels. Elements: `sound`

Table 16 *(continued)*

Name	Description
`encoding`	*Optional*. Corresponds to the **E** key for a sound object and is the encoding format for the sample data. Values are: — `raw (default)` — `signed` — `mulaw` — `alaw` Elements: `sound`
`rate`	*Required*. Corresponds to the **R** key for a sound object and is a real number describing the sampling rate, in samples per second. Elements: `sound`

Table 17 — Popup annotation attributes

Name	Description
`Open`	*Optional*. A flag specifying whether the annotation should initially be displayed open. Corresponds to the **Open** key in the pop-up annotation dictionary. Values are: — `yes` — `no` (default) Elements: `popup`

Table 18 — Link annotation attributes

Name	Description
`Highlight`	*Optional*. Corresponds to the **H** key in the link annotation dictionary. Describes the annotation's highlighting mode, the visual effect to be used when the mouse button is pressed or held down inside its active area. Values are: — `None` — `Invert` (default) — `Outline` — `Push` Elements: `link`

Table 18 *(continued)*

Name	Description
`coords`	*Optional*. Corresponds to the **QuadPoints** key in the link annotation dictionary. Value is one or more groups of eight comma-separated real numbers. Groups are separated by commas. An array of 8 x n numbers specifying the coordinates of n quadrilaterals in default user space. Each quadrilateral encompasses a word or group of contiguous words in the text underlying the annotation. The coordinates for each quadrilateral are given in the following order: `x1,y1,x2,y2,x3,y3,x4,y4` These specify the quadrilateral's four vertices in counter clockwise order. The text is oriented with respect to the edge connecting points `(x1,y1)` and `(x2,y2)`. Elements: `link`

Table 19 — Redaction annotation attributes

Name	Description
`coords`	*Optional*. Corresponds to the **QuadPoints** key in the redaction markup annotation dictionary. Value is one or more groups of eight comma-separated real numbers. Groups are separated by commas. An array of 8 x n numbers specifying the coordinates of n quadrilaterals in default user space. If present, these quadrilaterals denote the content region that is intended to be removed. If this entry is not present, the `rect` attribute denotes the content region that is intended to be removed. Elements: `redact`
`interior-color`	*Optional*. Corresponds to the **IC** key in the redaction annotation dictionary. Value is an array of three numbers in the range 0.0 to 1.0 specifying the components, in the DeviceRGB colour space, of the interior colour with which to fill the redacted region after the affected content has been removed. If this entry is absent, the interior of the redaction region is left transparent. Ignored if the `overlayappearance` element is present. Elements: `redact`
`overlay-text`	*Optional*. Corresponds to the **OverlayText** key in the redaction annotation dictionary. Value is a text string specifying the overlay text that should be drawn over the redacted region after the affected content has been removed. Ignored if `overlayappearance` is present. Elements: `redact`
`overlay-text-repeat`	*Optional*. Corresponds to the **Repeat** key in the redaction annotation dictionary. If true, then the text specified by `overlay-text` should be repeated to fill the redacted region after the affected content has been removed. Ignored if `overlayappearance` is present. Default value: false. Elements: `redact`
`justification`	*Optional*. The justification attribute corresponds to the **Q** key in the redaction annotation dictionary. It is ignored if `overlayappearance` is present. Values are: — `Left-justified` (default) — `Centered` — `Right-justified` Elements: `redact`

Table 20 — Border effect attributes

Name	Description
`intensity`	*Optional*. Default: `0` (meaning no effect). Corresponds to the **I** key in the border effect dictionary. A number describing the intensity of the effect. It is only considered valid when border effect style is set to `cloudy`. A higher value indicates more puffs in the cloud. Elements: `circle`, `polygon`, `polyline`, `square`
`style`	*Optional*. Default: `solid`. These values are appended to the list of style attribute values listed in Border style attributes. Values are: — `solid` — `cloudy` Elements: `circle`, `polygon`, `polyline`, `square`

Table 21 — Border style attributes

Name	Description
`width`	*Optional*. Default: `1` (meaning no effect). Value is a decimal number. Corresponds to the **W** key in the border style dictionary and specifies the border width in points. If this value is 0, no border is drawn. Elements: `circle`, `freetext`, `ink`, `line`, `polygon`, `polyline`, `square`, `text`
`dashes`	*Optional*. Default: `3`. Corresponds to the **D** key in the border style dictionary. A comma-separated list of numbers defining a pattern of dashes and gaps to be used in drawing a dashed border. The dash phase is not specified and is assumed to be 0. For example, a dashes attribute with value "3,2" specifies a border drawn with 3-point dashes alternating with 2-point gaps. Values are: One or more numbers separated by a comma. Example: — `3` — `3,5` — `4,3,2,3` Elements: `circle`, `freetext`, `ink`, `line`, `polygon`, `polyline`, `square`, `text`
`style`	*Optional*. These values are appended to the list of style attribute values listed in Border style attributes. Values are: — `solid` (default) — `dash` — `bevelled` — `inset` — `underline` Elements: `circle`, `freetext`, `ink`, `line`, `polygon`, `polyline`, `square`, `text`

Table 22 — Border array attributes

Name	Description
`HCornerRadius`	*Optional*. Corresponds to array index 0 in the **Border** key in the common annotation dictionary. The HCornerRadius is a number specifying the horizontal corner radius of the rectangular border in default user space units. Elements: `BorderStyleAlt`
`VCornerRadius`	*Optional*. Corresponds to array index 1 in the **Border** key in the common annotation dictionary. The `VCornerRadius` is a number specifying the vertical corner radius of the rectangular border in default user space units. Elements: `BorderStyleAlt`
`Width`	*Optional*. Corresponds to array index 2 in the **Border** key in the common annotation dictionary. The `Width` is a number specifying the width of the border in default user space units; if the `Width` is 0, no border is drawn. Elements: `BorderStyleAlt`
`DashPattern`	*Optional*. Corresponds to the optional dash array (array index 3) of the **Border** key in the common annotation dictionary. The `DashPattern` is a comma-separated list of numbers specifying the pattern of dashes and gaps of the border. Elements: `BorderStyleAlt`

Table 23 — Embedded file parameter attributes

Name	Description
`checksum`	*Optional*. Corresponds to the **CheckSum** key in the embedded file parameter dictionary. The checksum attribute is 16-byte string that is the checksum of the bytes of the uncompressed embedded file. The checksum is calculated by applying the standard MD5 message-digest algorithm to the bytes of the embedded file stream. Elements: `fileattachment`
`creation`	*Optional*. Value is a PDF date. The `creation` attribute corresponds to the **CreationDate** key in the embedded file parameter dictionary and is the date and time when the embedded file was created. Elements: `fileattachment`
`modification`	*Optional*. Value is in PDF date format. The modification attribute corresponds to the **ModDate** key in the embedded file parameter dictionary and is the date and time when the embedded file was last modified. Elements: `fileattachment`
`size`	*Optional*. The `size` attribute is an integer corresponding to the **Size** key in the embedded file parameter dictionary and is the size of the embedded file, in bytes. Elements: `fileattachment`

Table 24 — Stream attributes

Name	Description
`length`	*Required*. Corresponds to the **Length** key in the stream dictionary. Value is an integer describing the number of bytes in the stream. NOTE There may be an additional EOL marker, preceding **endstream**, which is not included in the count and is not logically part of the stream data. Elements: `data`, `resource`

Table 24 *(continued)*

Name	Description
`filter`	*Required*. Corresponds to the **Filter** key in the stream dictionary. The name of a filter to be applied in processing the stream data or comma-separated list of such names. Multiple filters should be specified in the order in which they are to be applied. Value is a single filter name or a list of names separated by commas. The filter name is a predefined value or user defined value. The predefined values are: — `ASCIIHexDecode` — `ASCII85Decode` — `LZWDecode` — `FlateDecode` — `RunLengthDecode` — `CCITTFaxDecode` — `JBIG2Decode` — `DCTDecode` — `JPXDecode` — `Crypt` Elements: `data`, `resource`

Table 25 — File specification attributes

Name	Description
`File`	*Required*. The file attribute corresponds to the **F** key in the file specification dictionary. Elements: `fileattachment`
`OriginalName`	*Required*. The `OriginalName` attribute corresponds to the **F** string in the remote go-to action and launch action dictionaries. Elements: `File`

Table 26 — Destination syntax attributes

Name	Description
`Name`	*Required*. The `Name` attribute specifies a named destination in the destination syntax allowing a destination to be referred to indirectly by means of a name object or a byte string. Elements: `Named`
`Page`	*Required*. Corresponds to the *page* object in the destination syntax. Elements: `Fit`, `FitB`, `FitBH`, `FitBV`, `FitH`, `FitR`, `FitV`, `XYZ`
`Left`	*Required*. Corresponds to the *left* object in the destination syntax. Elements: `FitBV`, `FitR`, `FitV`, `XYZ`
`Bottom`	*Required*. Corresponds to the *bottom* object in the destination syntax. Elements: `FitR`
`Right`	*Required*. Corresponds to the *right* object in the destination syntax. Elements: `FitR`

Table 26 *(continued)*

Name	Description
`Top`	*Required*. Corresponds to the *top* object in the destination syntax. Elements: `FitBH`, `FitH`, `FitR`, `XYZ`
`Zoom`	*Required*. Corresponds to the *zoom* object in the destination syntax. Elements: `XYZ`

Table 27 — Remote go-to attributes

Name	Description
`NewWindow`	*Optional*. Corresponds to the **NewWindow** key in the remote go-to action dictionary. Value is a flag specifying whether to open the destination document in a new window. If this flag is false, the destination document replaces the current document in the same window. If this entry is absent, the PDF processor should behave in accordance with the current user preference. Elements: `GoToR`

Table 28 — Launch attributes

Name	Description
`NewWindow`	*Optional*. Corresponds to the **NewWindow** key in the launch action dictionary. Value is a flag specifying whether to open the destination document in a new window. If this flag is false, the destination document replaces the current document in the same window. If this entry is absent, the PDF processor should behave in accordance with the current user preference. Elements: `Launch`

Table 29 — Named action attributes

Name	Description
`Name`	*Required*. The `Name` attribute corresponds to the **N** key in the named actions dictionary. Values are: — `NextPage` — `PrevPage` — `FirstPage` — `LastPage` Elements: `Named`

Table 30 — URI attributes

Name	Description
`Name`	*Required*. The `Name` attribute corresponds to the **URI** key in the URI action dictionary. Value is a string containing the uniform resource identifier to resolve, encoded in 7-bit ASCII. Elements: `URI`
`IsMap`	*Optional*. Default: `false` The `IsMap` attribute corresponds to the **IsMap** key in the action dictionary. Value is a flag specifying whether to track the mouse position when the URI is resolved. Elements: `URI`

Table 31 — Mac OS file information attributes

Name	Description
`Creator`	*Optional*. Corresponds to the **Creator** key in the Mac OS file information dictionary. Value is a string containing the embedded file's creator signature. Elements: `resource`
`Subtype`	*Optional*. Corresponds to the **Subtype** key in the Mac OS file information dictionary. Value is a string containing the embedded file's file type. Elements: `resource`
NOTE As of PDF 2.0, these attributes have been deprecated.	

Table 32 — Miscellaneous attributes

Name	Description
`mimetype`	*Optional*. Value is the subtype of the embedded file. The value of this entry shall be a first-class name, as defined in ISO 32000-2. Names without a registered prefix shall conform to the MIME media type names defined in Reference [2], with the provision that characters not allowed in PDF names shall use the 2-character hexadecimal code format described as Name Objects in ISO 32000-2. Elements: `fileattachment`
`mode`	*Required.* Values are: — `filtered` — `raw` Elements: `data`, `resource`
`encoding`	*Required*. The encoding format of the element content. Values are: — `ascii` — `hex` Elements: `data`, `resource`
NOTE The attributes below do not correspond to a PDF key.	

6.7 Mapping tables

6.7.1 PDF to XFDF

Table 33 provides the mapping that shall be used when taking keys from PDF and mapping them to entries in XFDF. The E/A entry indicates whether the key shall map to an element (E) or attribute (A). XFDF `data`, `encoding` and `mode` have no corresponding PDF key.

Table 33 — Mappings from PDF keys to XFDF elements and attributes

PDF key	Key value	Dictionary	XFDF	E/A
			`data`	E
			`encoding`	A
			`mode`	A
`A`		Link annotation	`Action`	E
`A`		Annotation	`OnActivation`	E
`AP`		Annotation	`appearance`	E
`Annots`		FDF	`annots`	E

Table 33 *(continued)*

PDF key	Key value	Dictionary	XFDF	E/A
B		Sound object	bits	A
Border		Annotation	DashPattern	A
Border		Annotation	HCornerRadius	A
Border		Annotation	VCornerRadius	A
Border		Annotation	Width	A
Border		Link annotation	BorderStyleAlt	E
bottom		Destination syntax annotation	Bottom	A
C		Annotation	color	A
C		Sound object	channels	A
CA		Markup annotation	opacity	A
Cap		Line annotation	caption	A
CheckSum		Embedded file parameter	checksum	A
CO		Line annotation	caption-offset-h	A
CO		Line annotation	caption-offset-v	A
Contents		Annotation	contents	E
CP		Line annotation	caption-style	A
CreationDate		Embedded file parameter	creation	A
CreationDate		Markup annotation	creationdate	A
Creator		Mac OS file information	creator	A
D		Border style	dashes	A
DA		Free text or caret annotation	defaultappearance	E
DA		Redaction annotation	defaultappearance	E
Dest		Link annotation	Dest	E
DS		Free text annotation	defaultstyle	E
E		Sound object	encoding	A
F		Annotation	flags	A
F		FDF	f	E
F		FDF	href	A
F		File specification	file	A
F		File specification	OriginalName	A
F		Remote go-to or launch annotation	File	A
Fields		FDF	field	
Fields		FDF	fields	E
Filter		Stream	filter	A
Fit		Destination syntax	Fit	E
FitB		Destination syntax	FitB	E
FitBH		Destination syntax	FitBH	E
FitBV		Destination syntax	FitBV	E
FitH		Destination syntax	FitH	E
FitR		Destination syntax	FitR	E
FitV		Destination syntax	FitV	E
GoTo		Action type	GoTo	E
GoToR		Action type	GoToR	E
H		Link annotation	Highlight	A

Table 33 *(continued)*

PDF key	Key value	Dictionary	XFDF	E/A
I		Border effect	intensity	A
IC		Redaction annotation	interior-color	A
IC		Square or circle annotation	interior-color	A
ID		FDF	ids	E
ID		FDF	modified	A
ID		FDF	original	A
InkList		Ink annotation	gesture	E
InkList		Ink annotation	inklist	E
IRT		Markup annotation	inreplyto	A
IsMap		Action dictionary annotation	IsMap	A
IT		Markup annotation	intent	A
L		Line annotation	end	A
L		Line annotation	start	A
Launch		Action type	Launch	E
LE		Line annotation	head	A
LE		Line annotation	tail	A
left		Destination syntax annotation	Left	A
Length		Stream	length	A
LL		Line annotation	leaderLength	A
LLE		Line annotation	leaderExtend	A
LLO		Line annotation	leader-offset	A
M		Annotation	date	A
ModDate		Embedded file parameter	modification	A
N		Named action	Name	A
NM		Annotation	name	A
Name		File attachment annotation	icon	A
Name		Rubber stamp annotation	icon	A
Name		Sound annotation	icon	A
Name		Text annotation	icon	A
Named		Action type	Named	E
NewWindow		Remote go-to action annotation	NewWindow	A
NewWindow		Launch parameter annotation	NewWindow	A
Open		Pop-up annotation	open	A
Page		FDF file	page	A
Q		Free text annotation	justification	A
Q		Redaction annotation	justification	A
QuadPoints		Text markup annotation	coords	A
QuadPoints		Link annotation	coords	A
QuadPoints		Redaction annotation	coords	A
R		Sound object	rate	A
RC		Markup annotation	contents-richtext	E
RD		Caret, square or circle annotation	fringe	A
RO		Redaction annotation	overlayappearance	E
RT		Markup annotation	replyType	A

Table 33 *(continued)*

PDF key	Key value	Dictionary	XFDF	E/A
RV		Fields containing variable text	value-richtext	E
Rect		Annotation	rect	A
Repeat		Redaction annotation	overlay-text-repeat	A
ResFork		Mac OS file information	resource	E
right		Destination syntax	Right	A
Root		FDF	xfdf	E
Rotate		Freetext and stamp annotations	rotation	A
S		Border style	style	A
Size		Embedded file parameter	size	A
State		Text annotation	state	A
StateModel		Text annotation	statemodel	A
Subj		Markup annotation	subject	A
Subtype	Caret	Annotation	caret	E
Subtype	Circle	Annotation	circle	E
Subtype	FileAttachment	Annotation	fileattachment	E
Subtype	FreeText	Annotation	freetext	E
Subtype	Highlight	Annotation	highlight	E
Subtype	Ink	Annotation	ink	E
Subtype	Line	Annotation	line	E
Subtype	Link	Annotation	link	E
Subtype	Polygon	Annotation	polygon	E
Subtype	Polyline	Annotation	polyline	E
Subtype	Popup	Annotation	popup	E
Subtype	Projection	Annotation	projection	E
Subtype	Redact	Annotation	redact	E
Subtype	Sound	Annotation	sound	E
Subtype	Square	Annotation	square	E
Subtype	Squiggly	Annotation	squiggly	E
Subtype	Stamp	Annotation	stamp	E
Subtype	StrikeOut	Annotation	strikeout	E
Subtype	Text	Annotation	text	E
Subtype	Underline	Annotation	underline	E
Subtype		Embedded file stream	mimetype	A
Subtype		Mac OS file information	subtype	A
Sy		Caret annotation	symbol	A
T		Annotation	title	A
T		FDF field	name	E
top		Destination syntax	Top	A
URI		URI action	Name	A
URI		Action type	URI	E
V		FDF field	value	E
Vertices		Polygon or polyline annotation	vertices	E
W		Border style	Width	A
XYZ		Destination syntax	XYZ	E

Table 33 *(continued)*

PDF key	Key value	Dictionary	XFDF	E/A
zoom		Destination syntax	Zoom	A

6.7.2 XFDF to PDF

Table 34 provides the mapping that shall be used when mapping XFDF element and attributes to PDF keys. The E/A entry indicates whether the XFDF entry is an element (E) or attribute (A).

Table 34 — Mappings from XFDF elements and attributes to PDF keys

XFDF	E/A	PDF key	Key value	Dictionary
Action	E	A		Link annotation
annots	E	Annots		FDF
appearance	E	AP		Annotation
bits	A	B		Sound object
BorderStyleAlt	E	Border		Link annotation
Bottom	A	bottom		Destination syntax annotation
caption	A	Cap		Line annotation
caption-offset-h	A	CO		Line annotation
caption-offset-v	A	CO		Line annotation
caption-style	A	CP		Line annotation
caret	E	Subtype	Caret	Annotation
channels	A	C		Sound object
checksum	A	CheckSum		Embedded file parameter
circle	E	Subtype	Circle	Annotation
color	A	C		Annotation
contents	E	Contents		Annotation
contents-richtext	E	RC		Markup annotation
coords	A	QuadPoints		Text markup annotation
coords	A	QuadPoints		Link annotation
coords	A	QuadPoints		Redaction annotation
creation	A	CreationDate		Embedded file parameter
creationdate	A	CreationDate		Markup annotation
creator	A	Creator		Mac OS file information
dashes	A	D		Border style
DashPattern	A	Border		Annotation
data	E			
date	A	M		Annotation
defaultappearance	E	DA		Free text annotation
defaultappearance	E	DA		Redaction annotation
defaultstyle	E	DS		Free text annotation
Dest	E	Dest		Link annotation
encoding	A			
encoding	A	E		Sound object
end	A	L		Line annotation
f	E	F		FDF
field	E	Fields		FDF

Table 34 *(continued)*

XFDF	E/A	PDF key	Key value	Dictionary
fields	E	Fields		FDF
file	A	F		File specification
File	E	F		Remote go-to or launch annotation
fileattachment	E	Subtype	FileAttachment	Annotation
filter	A	Filter		Stream
Fit	E	Fit		Destination syntax
FitB	E	FitB		Destination syntax
FitBH	E	FitBH		Destination syntax
FitBV	E	FitBV		Destination syntax
FitH	E	FitH		Destination syntax
FitR	E	FitR		Destination syntax
FitV	E	FitV		Destination syntax
flags	A	F		Annotation
freetext	E	Subtype	FreeText	Annotation
fringe	A	RD		Caret, square or circle annotation
gesture	E	InkList		Ink annotation
GoTo	E	GoTo		Action type
GoToR	E	GoToR		Action type
HCornerRadius	A	Border		Annotation
head	A	LE		Line annotation
Highlight	A	H		Link annotation
highlight	E	Subtype	Highlight	Annotation
href	A	F		FDF
icon	A	Name		File attachment annotation
icon	A	Name		Rubber stamp annotation
icon	A	Name		Sound annotation
icon	A	Name		Text annotation
ids	E	ID		FDF
ink	E	Subtype	Ink	Annotation
inklist	E	InkList		Ink annotation
inreplyto	A	IRT		Markup annotation
intensity	A	I		Border effect
intent	A	IT		Freetext, line or polyline annotation
interior-color	A	IC		Square or circle annotation
interior-color	A	IC		Redaction annotation
IsMap	A	IsMap		Action dictionary annotation
justification	A	Q		Free text annotation
justification	A	Q		Redaction annotation
Launch	E	Launch		Action type
leader-offset	A	LLO		Line annotation
leaderExtend	A	LLE		Line annotation
leaderLength	A	LL		Line annotation

Table 34 *(continued)*

XFDF	E/A	PDF key	Key value	Dictionary
Left	A	left		Destination syntax annotation
length	A	Length		Stream
line	E	Subtype	Line	Annotation
link	E	Subtype	Link	Annotation
mimetype	A	Subtype		Embedded file stream
mode	A			
modification	A	ModDate		Embedded file parameter
modified	A	ID		FDF
Name	A			Destination syntax
Name	A	N		Named action
Name	A	URI		URI action
name	A	NM		Annotation
name	E	T		FDF field
Named	E	Named		Action type
Named	E			Destination syntax
NewWindow	A	NewWindow		Remote go-to action annotation
NewWindow	A	NewWindow		Launch parameter annotation
OnActivation	E	A		Annotation
opacity	A	CA		Markup annotation
open	A	Open		Pop-up annotation
original	A	ID		FDF
OriginalName	A	F		File specification annotation
overlayappearance	E	RO		Redaction annotation
overlay-text	A	OverlayText		Redaction annotation
overlay-text-repeat	A	Repeat		Redaction annotation
Page	A	page		Destination syntax
page	A	Page		FDF file
polygon	E	Subtype	Polygon	Annotation
polyline	E	Subtype	Polyline	Annotation
popup	E	Subtype	Popup	Annotation
projection	E	Subtype	Projection	Annotation
rate	A	R		Sound object
rect	A	Rect		Annotation
redact	E	Subtype	Redact	Annotation
replyType	A	RT		Markup annotation
resource	E	ResFork		Mac OS file information
Right	A	right		Destination syntax
rotation	A	Rotate		Freetext and stamp annotations
size	A	Size		Embedded file parameter
sound	E	Subtype	Sound	Annotation
square	E	Subtype	Square	Annotation
squiggly	E	Subtype	Squiggly	Annotation
stamp	E	Subtype	Stamp	Annotation
start	A	L		Line annotation

Table 34 *(continued)*

XFDF	E/A	PDF key	Key value	Dictionary
state	A	State		Text annotation
statemodel	A	StateModel		Text annotation
strikeout	E	Subtype	StrikeOut	Annotation
style	A	S		Border style
subject	A	Subj		Markup annotation
subtype	A	Subtype		Mac OS file information
symbol	A	Sy		Caret annotation
tail	A	LE		Line annotation
text	E	Subtype	Text	Annotation
title	A	T		Annotation
Top	A	top		Destination syntax
underline	E	Subtype	Underline	Annotation
URI	E	URI		Action type
value	E	V		FDF field
value-richtext	E	RV		Fields containing variable text
VCornerRadius	A	Border		Annotation
vertices	E	Vertices		Polygon or polyline annotation
Width	A	Border		Annotation
width	A	W		Border style
xfdf	E	Root		FDF
XYZ	E	XYZ		Destination syntax
Zoom	A	zoom		Destination syntax

7 Reference for 3D and RichMedia annotations

7.1 General

Comments may also be placed on 3D or rich media annotations. A major difference between 3D or rich media annotations and other page annotations is in their ability to define structured content. For example, a 3D annotation may contain several views of a 3D model or a rich media annotation may contain a video sequence consisting of many frames. When a comment is placed on a 3D or rich media annotation, the comment is specific to a 3D view for the 3D page annotation or to a specific frame in a video for a rich media annotation.

Some additional context specific information needs to be kept with the comment in order to support the connection between a comment and the 3D or rich media annotation. When a comment is placed on 3D or rich media annotation, this extra information identifies both the 3D or rich media annotation and the view within that annotation that the comment is associated with. To support the ability to export and import these comments and to ensure that their association to a 3D or rich media annotation remains intact, this extra data content shall be part of the XFDF specification.

7.2 Various scenarios of comments on a 3D annotation

7.2.1 General

Let us consider three different situations where comments are placed on a 3D or rich media annotation. The first situation occurs when a 2D page comment is selected from the Comment & Markup menu and placed on top of a 3D view. In this case, the comment is expected to only appear when its associated 3D view is visible (or when the comment is selected from the comment or view pane). The second situation

occurs when a 2D page comment is selected from the Comment & Markup menu and placed on top of a rich media view. In this case, the comment is expected to only appear when its associated video frame is visible. The third situation occurs when a 3D comment or measurement is placed on a 3D view. In this case, the comment's markup is a part of the 3D scene definition and a projection annotation is used as a surrogate within the commenting system to represent the 3D comment.

In all three cases, comment's extra data contains the information to identify comment's associated 3D or rich media annotation and the view within that annotation. One complication that arises is that when a comment is placed on a 3D or rich media annotation, it is possible that a new 3D view is also created. If the user creates a new view with a comment and exports that comment, it is expected that when the comment is imported into a document where the view does not exist yet, the view will be created in the importing document. To support this, the view definition contained within the extra data structure should contain enough information to accurately recreate the original view.

7.2.2 Example of a comment on a 3D annotation

The following is an example XFDF file generated when a single 2D circle comment is placed on a 3D view and exported. The extra data-related tags are included in the example.

```
<?xml version="1.0" encoding="UTF-8"?>
<xfdf xmlns="http://ns.adobe.com/xfdf/" xml:space="preserve">
   <annots>
      <circle color="#FF0000" creationdate="D:20081106121144-
      08'00'" flags="print" date="D:20081106121144-08'00'"
      name="bc74fb1b-5044-4691-8b5a-af9de0923719" page="0"
      fringe="0.5,0.5,0.5,0.5"
      rect="230.045395,452.626088,374.557587,548.992188"
      subject="Oval" title="jahall">
         <ex_data subtype="Markup3D">
            <exdata3d>
               <anno3dname>3D1</anno3dname>
               <view3d>...</view3d>
            </exdata3d>
         </ex_data>
         <popup flags="print,nozoom,norotate" open="no" page="0"
         rect="612,428.492188,792,548.492188"/>
         </circle>
   </annots>
   <f href="vice_copy3.pdf"/>
   <ids original="D0F211E86F278A45A4C62BE7B0DF7C9F"
   modified="E6832FBBAA8B5840B0E075B9D59E71D4"/>
</xfdf>
```

The ex_data tag is a child of a specific type of annotation; in this case circle and has an attribute that defines the type of extra data contained within. The type of data contained is a function of the context in which the comment appears. In this example, the context type is “Markup3D” indicating that the content contains information identifying 3D annotation and defining the 3D view contents.

7.3 The ex_data annotation subelement

7.3.1 ex_data

7.3.1.1 Overview

The ex_data element is a child of an annotation element and serves as a container for extra context specific data elements. The ex_data element corresponds to the ExData key in the FDF dictionary. The ex_data annotation subelement is intended to be an open-ended hook for metadata inclusion that supports all present and future ExData subtypes. It is not specific to commenting on 3D annotations.

The ex_data subelement applies to all of the markup annotation types supported in XFDF. The various markup annotation types supported in XFDF are:

— Text

— FreeText

— Line

— Square

— Circle

— Polygon

— Polyline

— Highlight

— Underline

— Squiggly

— StrikeOut

— Stamp

— Caret

— Ink

— Popup

— FileAttachment

— Sound

— Projection

To ensure the security of imported XFDF data, no ex_data subelement nor any of its subelements, recursively, may refer to any external XML entity.

7.3.1.2 Content model

```
( exdata3d? | other_type_specific_content )
```

7.3.1.3 Attributes

ex_data attributes

`subtype` *Required*

The type of extra data contained corresponds to the PDF ExData dictionary's Subtype entry. Currently there are 4 subtypes supported:

— Markup3D – a 2D page comment placed on a 3D annotation.

— 3DM – a 3D comment or 3D measurement.

— RichMedia – a 2D page comment on a rich media annotation.

— GeoMarkup – a 2D measurement placed on 2D geospatial data.

NOTE The GeoMarkup type contains no extra data but is used as a tag, such that the application knows that the comment is associated to 2D geospatial data.

7.4 ex_data3d related elements

7.4.1 exdata3d

7.4.1.1 Overview

The exdata3d element is a child of an ex_data element and serves as the top-level container for data defining the connection between a comment and a 3D or rich media annotation. This element is valid for ex_data types: Markup3D, RichMedia and 3DM. The measurementcomment comment tag is only valid in the context of the 3DM type.

7.4.1.2 Content model

((anno3dname | md5checksum) & view3d & measurementcomment)*

7.4.1.3 Attributes

None.

7.4.2 anno3dname

7.4.2.1 Overview

The anno3dname element is a child of an exdata3d element and defines the name of the 3D or rich media annotation to which this comment is associated. The containing annotation element defines the page on which the comment is present; therefore, annotation names only need to be unique within a single page. The anno3dname matches the NM key in the annotations dictionary.

7.4.2.2 Content model

Text String.

7.4.2.3 Attributes

None.

7.4.3 md5checksum

7.4.3.1 Overview

Note that all annotations are not named; the NM key is optional. If the annotation is not named, the md5checksum tag is used to identify the appropriate 3D or rich media annotation. The md5checksum is the checksum of the data associated with the 3D or rich media annotation. The md5checksum element is a child of an exdata3d element.

7.4.3.2 Content model

Text String.

7.4.3.3 Attributes

None.

7.4.4 measurename

7.4.4.1 Overview

For 3D comments and measurements, the specific 3D markup within the view also needs to be identified. The view has a list of named measurements (or 3D comments); this tag will match one of the names on the view's list. This tag will only be present if the ex_data type is 3DM.

7.4.4.2 Content model

Text String.

7.4.4.3 Attributes

None.

7.5 The view3d related elements

7.5.1 General

For 3D annotations, a 3D view defines the set of parameters that will create a specific presentation of a collection of 3D scene geometry.

The following is an example of a 3D view definition.

```
<view3d>
   <externalname>SectionView5</externalname>
   <internalname>60abf029-e84c-4969-8359-18f1639c6077</internalname>
   <cameraxform>
      -0.493431 0.869785 0.000000 0.000000
      0.307474 0.174430 0.935433 0.000000
      0.813625 0.461571 -0.353505 0.000000
      -0.356615 -0.287046 0.181059 1.000000
   </cameraxform>
```

```
    <targetdistance>0.434344</targetdistance>
    <projection projtype="perspective">
       <fieldofview>30.000001</fieldofview>
       <scaletype>min</scaletype>
    </projection>
    <background entirebackground="true">0.937256 0.937256
    0.937256</background>
    <renderinginfo mode="solid"/>
    <lighting scheme="headlamp"/>
    <shouldresetnodes/>
</view3d>
```

7.5.2 view3d

7.5.2.1 Overview

The view3d tag is the top container for the data required to define the 3D view that a comment is associated with.

7.5.2.2 Content model

(externalname & internalname & (cameraxform | u3dmatrixsource) & targetdistance & projection & clip & background & renderinginfo & lighting & crosssection? & nodeparameter & shouldresetnodes? & measure* & stateinfo* & snapshot?)*

7.5.2.3 Attributes

None.

7.5.3 externalname

7.5.3.1 Overview

The externalname tag defines the user visible name for the view.

7.5.3.2 Content model

Text String.

7.5.3.3 Attributes

None.

7.6 Camera related elements

7.6.1 General

The camera transformation associated with a view can be specified by two methods. The first is by directly specifying the transformation matrix and the second is by passing the Id of a node in the U3D

scene that contains a transformation matrix. For further information, refer to the PDF specification's 3D view definition (see ISO 32000-2:2017, 13.6.4).

7.6.2 cameraxform

7.6.2.1 Overview

The cameraxform tag defines a 12 element 3D transformation matrix that specifies the position and orientation of the camera in world coordinates.

7.6.2.2 Content model

A list of 12 doubles.

7.6.2.3 Attributes

None.

7.6.3 u3dmatrixsource

7.6.3.1 Overview

The u3dmatrixsource tag defines the name of a node in the U3D scene graph that contains a view transformation.

7.6.3.2 Content model

Text String.

7.6.3.3 Attributes

None.

7.6.4 targetdistance

7.6.4.1 Overview

The targetdistance tag defines a non-negative number indicating a distance in the camera coordinate system along the z-axis to the centre of orbit for this view.

7.6.4.2 Content model

Double.

7.6.4.3 Attributes

None.

7.7 View projection related elements

7.7.1 projection

7.7.1.1 Overview

The projection tag is a container for which entries define the mapping of 3D camera coordinates onto the target coordinate system of the annotation.

7.7.1.2 Content model

```
((fieldofview & (scalevalue | scaletype)? ) | (viewplanesize? & scaletype?))
```

7.7.1.3 Attributes

Launch attributes

`projtype` *Required*

The type of view projection:

— parallel

— *perspective*

7.7.2 fieldofview

7.7.2.1 Overview

The fieldofview tag is valid only for perspective projections. It defines a number between 0 and 180, inclusive, specifying the field of view of the virtual camera, in degrees. It defines a cone in 3D space centred around the z-axis and a circle where the cone intersects the near clipping plane. The circle, along with the value formed from either the scalevalue or the scaletype, specify the scaling of the projected artwork when rendered in the 2D plane of the annotation.

7.7.2.2 Content model

Double.

7.7.2.3 Attributes

None.

7.7.3 viewplanesize

7.7.3.1 Overview

The viewplanesize tag is valid only for parallel projections. It defines a positive number that specifies the scale factor to be applied to both the x and y coordinates when projecting onto the annotation's target coordinate system.

7.7.3.2 Content model

Double.

7.7.3.3 Attributes

None.

7.7.4 scalevalue

7.7.4.1 Overview

The scalevalue tag is only valid for perspective projections. It defines the diameter of the circle formed by the intersection of the near plane and the cone specified by fieldofview. For parallel projections, it defines a positive number that specifies the scale factor to be applied to both the x and y coordinates when projecting onto the annotation's target coordinate system.

7.7.4.2 Content model

Double.

7.7.4.3 Attributes

None.

7.7.5 scaletype

7.7.5.1 Overview

The scaletype tag contains a name that specifies a strategy for binding (scaling to fit) the near plane's x and y coordinates onto the annotation's target coordinate system. For parallel projections, the scaling specified in this entry is applied in addition to the scaling factor specified by the viewplanesize entry.

7.7.5.2 Content model

Text string for which value may be one of the following:

— *W - Scale to fit the width of the annotation*

— *H - Scale to fit the height of the annotation*

— *Min - Scale to fit the lesser of width or height of the annotation*

— *Max - Scale to fit the greater of width or height of the annotation*

— *Absolute - No scaling should occur due to binding (parallel projections only)*

7.7.5.3 Attributes

None.

7.7.6 clip

7.7.6.1 Overview

The clip tag is present when near and far clipping planes need to be applied to the view.

7.7.6.2 Content model

None.

7.7.6.3 Attributes

clip attributes

Near	*Required* for perspective projections; *optional* for parallel projections. The near clipping distance, expressed in the camera coordinate system. No parts of objects for which z-coordinates are less than the value of this entry are drawn. For perspective projections, the value shall be positive; for parallel projections, the value shall be non-negative and the default value is 0.
Far	*Optional*. The far clipping distance, expressed in the camera coordinate system. No parts of objects for which z-coordinates are greater than the value of this entry are drawn. If this entry is absent, no far clipping occurs.

7.8 View background related elements

7.8.1 background

7.8.1.1 Overview

The background tag contains information about the how the view background should be rendered.

7.8.1.2 Content model

(color)

7.8.1.3 Attributes

background attributes

`entirebackground`	*Optional.* If true, the background should apply to the entire annotation; if false, the background should apply only to the rectangle specified by the annotation's 3D view box. The default value is false.

7.8.2 color

7.8.2.1 Overview

The color tag defines a 3 element list representing the RGB colour for the background.

7.8.2.2 Content model

A list of 3 doubles.

7.8.2.3 Attributes

None.

7.9 Model rendering related elements

7.9.1 renderinginfo

7.9.1.1 Overview

The renderinginfo tag contains information about how the view geometry should be rendered.

7.9.1.2 Content model

(auxcolor? & facecolor? & opacity? & creasevalue?)

7.9.1.3 Attributes

renderinginfo attributes

`mode` *Required*

— solid – displays textured and lit geometric shapes.

— solidwireframe – displays textured and lit geometric shapes (triangles) with single colour edges on top of them. The colour of these edges is determined by the auxcolor entry.

— transparent – displays textured and lit geometric shapes (triangles) with an added level of transparency.

— transparentwireframe – displays textured and lit geometric shapes (triangles) with an added level of transparency, with single colour opaque edges on top of it. The colour of these edges is determined by the auxcolor entry.

— boundingbox – displays the bounding box edges of each node, aligned with the axes of the local coordinate space for that node. The colour of the bounding box edges is determined by the auxcolor entry.

— transparentboundingbox – displays bounding boxes faces of each node, aligned with the axes of the local coordinate space for that node, with an added level of transparency. The colour of the bounding box faces is determined by the facecolor entry.

— transparentboundingboxoutline – displays bounding boxes edges and faces of each node, aligned with the axes of the local coordinate space for that node, with an added level of transparency. The colour of the bounding box edges is determined by the auxcolor entry. The colour of the bounding boxes faces is determined by the facecolor entry.

— wireframe – displays only edges in a single colour. The colour of these edges is determined by the auxcolor entry.

— shadedwireframe – displays only edges; although interpolates their colour between their two vertices and applies lighting.

— hiddenwireframe – displays edges in a single colour; although removes back-facing and obscured edges. The colour of these edges is determined by the auxcolor entry.

— vertices displays – only vertices in a single colour. The colour of these points is determined by the auxcolor entry.

— shadedvertices – displays only vertices; although uses their vertex colour and applies lighting.

— illustration – displays silhouette edges with surfaces and removes obscured lines. The colour of these edges is determined by the auxcolor entry and the colour of the surfaces is determined by the facecolor entry.

— solidoutline – displays silhouette edges with lit and textured surfaces and removes obscured lines. The colour of these edges is determined by the auxcolor entry.

— shadedillustration – *displays silhouette edges with lit and textured surfaces and an additional emissive term to remove poorly lit areas of the artwork. The colour of these edges is determined by the auxcolor entry.*

7.9.2 auxcolor

7.9.2.1 Overview

The auxcolor tag defines a 3 element list representing the RGB used in many of the rendering modes.

7.9.2.2 Content model

A list of 3 doubles.

7.9.2.3 Attributes

None.

7.9.3 facecolor

7.9.3.1 Overview

The facecolor tag defines a 3 element list representing the RGB colour used for illustration and transparent bounding box outline rendering modes.

7.9.3.2 Content model

A list of 3 doubles.

7.9.3.3 Attributes

None.

7.9.4 opacity

7.9.4.1 Overview

The opacity tag defines a number specifying the opacity applied by some render modes, using a standard additive blend. The default value is 0,5.

7.9.4.2 Content model

Double.

7.9.4.3 Attributes

None.

7.9.5 creasevalue

7.9.5.1 Overview

The creasevalue tag contains a number specifying the angle, in degrees, to be used as the crease value for determining silhouette edges. If two front-facing faces share an edge and the angle between the normals of those faces is greater than or equal to the crease value, the shared edge is considered to be a silhouette edge. The default value is 45.

7.9.5.2 Content model

Double.

7.9.5.3 Attributes

None.

7.10 Lighting related elements

7.10.1 lighting

7.10.1.1 Overview

The lighting tag specifies the lighting scheme to be used when rendering 3D artwork with this view.

7.10.1.2 Content model

None.

7.10.1.3 Attributes

lighting attributes

`mode` *Required*

— artwork - lights as specified in the 3D artwork

— none - no lights are used

— white - three blue-grey infinite lights, no ambient term

Light 1 Colour: < 0.38, 0.38, 0.45 > Direction: < –2.0, –1.5, –0.5 >
Light 2 Colour: < 0.6, 0.6, 0.67 > Direction: < 2.0, 1.1, –2.5 >
Light 3 Colour: < 0.5, 0.5, 0.57 > Direction: < –0.5, 0.0, 2.0 >

— day - three light-grey infinite lights, no ambient term

Light 1 Colour: < 0.5, 0.5, 0.5 > Direction: < –2.0, –1.5, –.5 >
Light 2 Colour: < 0.8, 0.8, 0.9 > Direction: < 2.0, 1.1, –2.5 >
Light 3 Colour: < 0.9, 0.9, 0.9 > Direction: < 0.02, 0.01, 2.0 >

— night - one yellow, one aqua and one blue infinite light, no ambient term

Light 1 Colour: < 1, .75, .39 > Direction: < –2.0, –1.5, –0.5 >
Light 2 Colour: < 0.31, 0.47, 0.55 > Direction: < 2.0, 1.1, –2.5 >
Light 3 Colour: < .5, .5, 1.0 > Direction: < 0.0, 0.0, 2.0 >

— hard - three grey infinite lights, moderate ambient term

Light 1 Colour: < 0.5, 0.5, 0.5 > Direction: < -1.5, –1.5, –1.5 >
Light 2 Colour: < 0.8, 0.8, 0.9 > Direction: < 1.5, 1.5, –1.5 >
Light 3 Colour: < 0.9, 0.9, 0.9 > Direction: < –0.5, 0, 2.0 >
Ambient Colour: < 0.5, 0.5, 0.5 >

— primary - one red, one green and one blue infinite light, no ambient term

Light 1 Colour: < 1, 0.2, 0.5 > Direction: < –2.0, –1.5, –0.5 >
Light 2 Colour: < 0.2, 1.0, 0.5 > Direction: < 2.0, 1.1, –2.5 >
Light 3 Colour: < 0, 0, 1 > Direction: < 0.0, 0.0, 2.0 >

— blue - three blue infinite lights, no ambient term

Light 1 Colour: < 0.4, 0.4, 0.7 > Direction: < –2.0, –1.5, –0.5 >
Light 2 Colour: < 0.75, 0.75, 0.95 > Direction: < 2.0, 1.1, -2.5 >
Light 3 Colour: < 0.7, 0.7, 0.95 > Direction: < 0.0, 0.0, 2.0 >

— red - three red infinite lights, no ambient term

Light 1 Colour: < 0.8, 0.3, 0.4 > Direction: < -2.0, –1.5, –0.5 >
Light 2 Colour: < 0.95, 0.5, 0.7 > Direction: < 2.0, 1.1, –2.5 >
Light 3 Colour: < 0.95, 0.4, 0.5 > Direction: < 0.0, 0.0, 2.0 >

— cube - six grey infinite lights aligned with the major axes, no ambient term

Light 1 Colour: < .4, .4, .4 > Direction: < 1.0, 0.01, 0.01 >
Light 2 Colour: < .4, .4, .4 > Direction: < 0.01, 1.0, 0.01 >
Light 3 Colour: < .4, .4, .4 > Direction: < 0.01, 0.01, 1.0 >
Light 4 Colour: < .4, .4, .4 > Direction: < –1.0, 0.01, 0.01 >
Light 5 Colour: < .4, .4, .4 > Direction: < 0.01, –1.0, 0.01 >
Light 6 Colour: < .4, .4, .4 > Direction: < 0.01, 0.01, –1.0 >

— cad - three grey infinite lights and one light attached to the camera, no ambient term

Light 1 Colour: < 0.72, 0.72, 0.81 > Direction: < 0.0, 0.0, 0.0 >
Light 2 Colour: < 0.2, 0.2, 0.2 > Direction: < -2.0, –1.5, –0.5 >
Light 3 Colour: < 0.32, 0.32, 0.32 > Direction: < 2.0, 1.1, –2.5 >
Light 4 Colour: < 0.36, 0.36, 0.36 > Direction: < 0.04, 0.01, 2.0 >

— headlamp - single infinite light attached to the camera, low ambient term

Light 1 Colour: < 0.8, 0.8, 0.9 > Direction: < 0.0, 0.0, 0.0 >
Ambient Colour: < 0.1, 0.1, 0.1 >

7.11 Cross section related elements

7.11.1 General

The following is an example of a cross section definition.

```
<crosssection>
   <centerofrotation>0.000000 -0.093300 0.026100</centerofrotation>
   <planetilt1>0.000000</planetilt1>
   <planetilt2>0.000000</planetilt2>
```

```
      <alignment axis="z"/>
      <intersectionsvisible>
         <planecolor>0.999985 0.000000 0.000000</planecolor>
      </intersectionsvisible>
   </crosssection>
```

7.11.2 crosssection

7.11.2.1 Overview

The crosssection tag is a container for which entries define the parameters for a section plane operation to be applied to the view.

7.11.2.2 Content model

(centerofrotation? & planetilt1, planetilt2 & alignment & planevisible? & intersectionsvisible?)

7.11.2.3 Attributes

crosssection attributes

planeflip	Optional If true, the geometry on the negative side of the section plane (w.r.t. the plane normal) will be clipped, otherwise geometry on the positive side of the section plane will be clipped.
showtransparent	*Required* If true, the clipped geometry will be rendered in a transparent rendering mode.

7.11.3 centerofrotation

7.11.3.1 Overview

The centerofrotation tag contains three doubles specifying the (x, y, z) position of the centre of rotation on the cutting plane in world space coordinates.

7.11.3.2 Content model

A list of three doubles.

7.11.3.3 Attributes

None.

7.11.4 planetilt1, planetilt2

7.11.4.1 Overview

The section plane orientation is defined by specifying two angles (planetilt1, planetilt2) in degrees relative to either the x, y or z axis.

7.11.4.2 Content model

Double.

7.11.4.3 Attributes

None.

7.11.5 alignment

7.11.5.1 Overview

The axis to which the tilt angles are relative.

7.11.5.2 Content model

Text string; valid values are x, y or z.

7.11.5.3 Attributes

None.

7.11.6 planevisible

7.11.6.1 Overview

The planevisible tag, when present, indicates that the section plane should be visible in the 3D scene. The contents of this tag control the planes appearance.

7.11.6.2 Content model

(planecolor? & planeopacity?)

7.11.6.3 Attributes

None.

7.11.7 planecolor

7.11.7.1 Overview

The planecolor tag defines a 3 element list representing the RGB colour used in rendering the section plane.

7.11.7.2 Content model

A list of three doubles.

7.11.7.3 Attributes

None.

7.11.8 planeopacity

7.11.8.1 Overview

The planeopacity tag defines a number specifying the opacity of the section plane. The default value is 0,5.

7.11.8.2 Content model

Double.

7.11.8.3 Attributes

None.

7.11.9 intersectionsvisible

7.11.9.1 Overview

The intersectionsvisible tag, when present, indicates that the lines of intersection between the scene geometry and the section plane should be visible in the 3D scene. The contents of this tag control the planes appearance.

7.11.9.2 Content model

(intersectioncolor?)

7.11.9.3 Attributes

None.

7.11.10 intersectioncolor

7.11.10.1 Overview

The intersectioncolor tag defines a 3 element list representing the RGB colour used in rendering the lines of intersection between the scene geometry and the section plane.

7.11.10.2 Content model

A list of 3 doubles.

7.11.10.3 Attributes

None.

7.12 View specific node control related elements

7.12.1 nodeparameter

7.12.1.1 Overview

The nodeparameter tag is a container for view specific attributes to be applied to specific nodes in a 3D scene. Note that parent view3d tag can contain many nodeparameter tags, defining parameters for different nodes in the scene.

7.12.1.2 Content model

```
(nodeid & nodexform? & opacity? & noderendermode?)
```

7.12.1.3 Attributes

nodeparameter attributes

`visible` *Required*

Boolean value defining the node's visibility.

7.12.2 nodeid

7.12.2.1 Overview

The nodeid tag defines the name of a node in the scene graph that the node parameters are to be applied to.

7.12.2.2 Content model

Text String.

7.12.2.3 Attributes

None.

7.12.3 nodexform

7.12.3.1 Overview

The nodexform tag defines a transformation matrix that is to be applied to the named node in the scene graph.

7.12.3.2 Content model

A list of 12 doubles.

7.12.3.3 Attributes

None.

7.12.4 opacity

7.12.4.1 Overview

The opacity tag defines an opacity value that is to be applied to the named node in the scene graph.

7.12.4.2 Content model

Double.

7.12.4.3 Attributes

None.

7.12.5 noderendermode

7.12.5.1 Overview

The noderendermode tag defines a render mode value that is to be applied to the named node in the scene graph.

7.12.5.2 Content model

Text String is constrained to the values listed in the renderinginfo tag.

"solid", "solidwireframe", "transparent", "transparentwireframe", "boundingbox", "transparentboundingbox", "transparentboundingboxoutline", "wireframe", "shadedwireframe", "hiddenwireframe", "vertices", "shadedvertices", "illustration", "solidoutline", "shadedillustration"

7.12.5.3 Attributes

None.

7.13 Rich Media related elements

7.13.1 stateinfo

7.13.1.1 Overview

The stateinfo tag contains a text string containing state data to be passed to a rich media instance (usually a player) when the view is triggered. For example, an instance might contain a string that when sent to the player will move to a specific time with a video.

7.13.1.2 Content model

Text String.

7.13.1.3 Attributes

stateinfo attributes

`instance` *Required*

Text String; the name of the rich media instance.

7.13.2 snapshot

7.13.2.1 Overview

The snapshot tag contains an image that is displayed when the view is activated. If this tag is present, the image it describes is displayed in place of the current artwork.

7.13.2.2 Content model

Text String; the image is defined as base 64 encoded string.

7.13.2.3 Attributes

None.

7.14 Measurement related elements

7.14.1 measure

7.14.1.1 Overview

The measure tag is the top level container for all 3D measurement and comment instances. A 3D view may contain a list of one or more 3D measurements or 3D comments that are associated with the view content, so there may be many instances of the measure tag within a parent view3d tag.

7.14.1.2 Content model

```
(measurename & (linearmarkup | perpendicularmarkup |angularmarkup | radialmarkup | comment3dmarkup))
```

7.14.1.3 Attributes

```
None.
```

7.14.2 measurename

7.14.2.1 Overview

The measurename tag defines the user visible name for the 3D measurement or comment.

7.14.2.2 Content model

```
Text String.
```

7.14.2.3 Attributes

```
None.
```

7.15 Linear dimension related elements

7.15.1 General

A linear measurement is a markup showing the distance between two arbitrary points on a 3D model. An example is displayed in Figure 9.

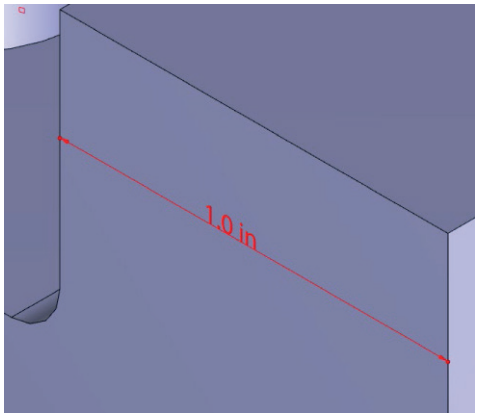
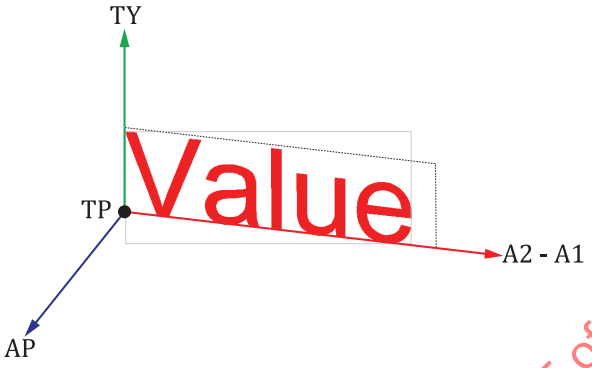


Figure 9 — Example of a measurement on a 3D model

As displayed in Figure 9, a 3D linear measurement consists of two filled circles (called anchor points), one at each of the two positions being measured, and a line with an arrowhead on each end connecting the two anchor points (referred to as the measure line). The measure line is labelled with a value representing the distance between the two anchor points.

As displayed in Figure 10, the value text is drawn on the Annotation Plane (AP) where the horizontal text path is defined by the vector from Anchor Point 1 (A1) to Anchor Point 2 (A2) with the text up direction defined by the vector TY. The lower left corner of the text box is positioned at the text anchor point (TP).



Key

AP annotation plane normal

Figure 10 — Example annotation text

The actual Y axis is formed by taking the cross product of AP and (A2 – A1); the vector TY is only used to determine the orientation of this Y axis.

If the text position TP is outside the area between A1 and A2, an extension line collinear to the measure line connecting TP to the closest anchor point is generated.

There are three parts to the text string displayed with the measurement; a numeric value, a units string and an optional user string. The display of the numeric value field number is also controlled by the precision value, which indicates how many digits to the right of the decimal point should be displayed. The viewer can convert the numeric value to a string and combine it with the units string and user text as appropriate.

The following is an example of a linear dimension.

```
<measure>
   <measurename>Measurement1</measurename>
   <linearmarkup>
      <annoplane> -0.981099 0.122982 0.149397</annoplane>
      <anchor1>-0.051400 -0.019600 0.052200</anchor1>
      <anchor1partname>Base</anchor1partname>
      <anchor2>-0.050300 -0.065700 0.052100</anchor2>
      <anchor2partname>Base</anchor2partname>
      <textposition>-0.050994 -0.036613 0.052163</textposition>
      <textydirection>0.150717 0.001452 0.988576</textydirection>
```

```
        <textsize>20.000000</textsize>
        <markupcolor>0.000000 1.000000 0.000000</markupcolor>
        <value>0.056113</value>
        <units>cm</units>
        <precision>3</precision>
      </linearmarkup>
  </measure>
```

7.15.2 Linearmarkup

7.15.2.1 Overview

The linearmarkup tag is a container that holds the data defining an instance of a point-to-point measurement.

7.15.2.2 Content model

(annoplane & anchor1 & anchor1partname? & anchor2 & anchor2partname? & textposition & textydirection & textsize? & markupcolor? & value & units & precision? & usertext?)

7.15.2.3 Attributes

None.

7.15.3 annoplane

7.15.3.1 Overview

The annoplane tag contains a list of three doubles defining the normal for the 3D annotation plane on which the measurement markup will lie.

7.15.3.2 Content model

A list of three doubles.

7.15.3.3 Attributes

None.

7.15.4 anchor1

7.15.4.1 Overview

The anchor1 tag contains a list of three doubles defining the model space position of the first anchor point in world space. It is assumed that this is a position on the 3D model associated with this view.

7.15.4.2 Content model

A list of three doubles.

7.15.4.3 Attributes

None.

7.15.5 anchor1partname

7.15.5.1 Overview

The anchor1partname tag defines the node name for the part that the point anchor1 lies on.

7.15.5.2 Content model

Text String.

7.15.5.3 Attributes

None.

7.15.6 anchor2

7.15.6.1 Overview

The anchor2 tag contains a list of three doubles defining the model space position of the second anchor point in world space. It is assumed that this is a position on the 3D model associated with this view.

7.15.6.2 Content model

A list of three doubles.

7.15.6.3 Attributes

None.

7.15.7 anchor2partname

7.15.7.1 Overview

The anchor2partname tag defines the node name for the part that the point anchor2 lies on.

7.15.7.2 Content model

Text String.

7.15.7.3 Attributes

None.

7.15.8 textposition

7.15.8.1 Overview

The textposition tag contains a list of three doubles defining the model space position of the text anchor (the start of the text string) associated with the measurement.

7.15.8.2 Content model

A list of three doubles.

7.15.8.3 Attributes

None.

7.15.9 textydirection

7.15.9.1 Overview

The textydirection tag contains a list of three doubles defining the up direction vector for the text string presenting the measurement value.

7.15.9.2 Content model

A list of three doubles.

7.15.9.3 Attributes

None.

7.15.10 textsize

7.15.10.1 Overview

The textsize tag contains the size for the text string presenting the measurement value. Note that the text associated with 3D measurements and comments scales with the view, so the text size is defined as the size in inches when the view is first activated.

7.15.10.2 Content model

Double.

7.15.10.3 Attributes

None.

7.15.11 markupcolor

7.15.11.1 Overview

The markupcolor tag contains a list of three doubles defining the RGB colour used in presenting the measurement markup.

7.15.11.2 Content model

A list of three doubles.

7.15.11.3 Attributes

None.

7.15.12 value

7.15.12.1 Overview

The value tag defines the numerical value representing a measurement value. This value is converted to a text string and displayed as part of the measurement text string.

7.15.12.2 Content model

Double.

7.15.12.3 Attributes

None.

7.15.13 units

7.15.13.1 Overview

The units tag contains a text string indicating the units of the associated measure value. This is appended to the value string when presenting the measurement.

7.15.13.2 Content model

Text String.

7.15.13.3 Attributes

None.

7.15.14 precision

7.15.14.1 Overview

The precision tag defines the number of decimal digits shown for the measurement value. If not specified, the default is three decimal places.

7.15.14.2 Content model

Integer.

7.15.14.3 Attributes

None.

7.15.15 usertext

7.15.15.1 Overview

The usertext tag contains an optional text string that is appended to the end of the measurement.

7.15.15.2 Content model

Text String.